A Hybrid Model of Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) Algorithm for Test Case Optimization

Abraham Kiran Joseph\textsuperscript{a}, Dr. G. Radhamani\textsuperscript{b} \textsuperscript{*}

\textsuperscript{a}Research Scholar, Dr.G.R Damodaran College of Science, Affiliated to Bharathiar University, Tamilnadu. E-mail: abrahamkiran@ymail.com

\textsuperscript{b}Professor/Director - Department of Computer Science, Dr.G.R Damodaran College of Science, Affiliated to Bharathiar University

Abstract

In this paper a hybrid model called Particle Swarm Artificial Bee Colony algorithm (PSABC) has been proposed. The PSABC algorithm is a combination of Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) Algorithm. PSABC has been used in this context to optimize the fitness value of a population in ABC algorithm using Particle Swarm Optimization. PSABC could be used to optimize the test suite, and integrate both ABC and PSO for a better result, as compared to their individual performance in terms of test case optimization. In PSABC, the sites are comparable to nodes in the Software under Test (SUT). While relating to the ABC algorithm, the bees here are the Test cases that adapt with time. The main objective of the bee is to find the areas with highest coverage and highest usage, so that failures could be identified at an earlier stage. The artificial bees adapt the test cases with time and the bee’s plan is to find out the areas where the nodes have higher coverage. The ABC algorithm is used to generate the optimal number of test-cases, which are sufficient to cover the paths. These paths are generated by using the Control Flow Graph (CFG), and the PSO generates the individual test cases first. This algorithm determines the node with the highest usage by a given test case. Based on the proposed hybrid approach, an optimal result for test case execution is obtained. The performance of the proposed method is evaluated and is compared with other optimization techniques such as PSO and Ant Colony Optimization (ACO).

Keywords: Search Domain based PSABC algorithm, Test cases, statement coverage and fault coverage, Optimization

1. Introduction

Software testing is one of the primary techniques used to achieve high quality in Software. Software testing is a time consuming and costly task it uses approximately 50\% of the software system development resources [1]. Software testing can also be defined as the process of verifying and evaluating it to make sure that software meets the technical and business requirements [2].

The maintenance phase in a Software Development Life Cycle involves regression testing to be done extensively. It is necessary to retest the existing test suite whenever any alterations are done to the software. Regression testing is the phenomenon of re-running the test cases from the test suite to assure error free and modified software. It guarantees that modifications in the software have not influenced its functional characteristics [3].

Verification is done to ensure that the software meets specification and is close to structural testing whereas validation is close to the functional testing and is done by executing the Software under Test (SUT). Broadly, testing techniques include functional (black box) and structural (white box) testing. Functional testing is based on functional requirements whereas structural testing is done on code itself. Gray box testing is a hybrid of white box testing and black box testing.

The main purpose of testing can be quality assurance, reliability estimation, validation or verification. The other objectives or software testing includes, [4]

\begin{itemize}
  \item The better it works the more efficiently it can be tested.
  \item Better the software can be controlled more the testing can be automated and optimized.
  \item The fewer the changes, the fewer the disruption to testing.
\end{itemize}

* Corresponding author. E-mail: abrahamkiran@ymail.com

266
• A successful test is the one that uncovers an undiscovered error.

Optimization techniques have been effectively used in test case generation and prioritization in recent years. Although, a number of optimization techniques had been proposed and good results have been obtained, problems such as complexity in dynamic data sets, and higher time consumption for convergence exist in the traditional optimization techniques. Thus, there is a scope of improvement for the improvement of optimization results. This research work focuses on using the appropriate optimization techniques, for test case prioritization which provides the optimal results.

This approach uses swarm intelligence based techniques for test case optimization using test case prioritization. A number of swarm intelligence approaches have been observed to produce significant results in terms of accuracy, convergence behaviour, and time taken. This research uses a couple of recent swarm intelligence approach called as the Particle Swarm Algorithm (PSO) and Artificial Bee Colony (ABC) Algorithm for Test Case Optimization [5].

1.1. Particle Swarm Optimization

PSO algorithm is motivated by the social behaviour of a collection of migrating birds trying to arrive an unknown destination. In PSO, each solution is a ‘bird’ in the flock and is known as a ‘particle’. A particle is equivalent to a chromosome (population member) in [6]. In [7], unlike Genetic Algorithm’s (GA’s), the evolutionary process in the PSO won’t generate new birds from parent ones. Instead, the birds in the population merely develop their social behaviour and as a result their movement towards a destination. The process is initiated with a collection of random particles (solutions), N. The ith particle is denoted by its position as a point in S-dimensional space, where S denotes the number of variables.

1.2. Artificial Bee Colony

In ABC algorithm, the solution of the optimization problem is denoted by the position of a food source and the quality of the solution is represented by the nectar amount of the source. In the initial step of ABC, the locations for the food source are generated randomly. In other words, for SN (the number of employed or onlooker bees) solutions, a randomly distributed initial population is produced. In the solution space, each solution $X_i = (x_{i1}, x_{i2}, \ldots, x_{iS})$ is a vector on the scale of its optimization parameters Y [8].

2. Related Work

Ant Colony Optimization (ACO) is a technique based on the real life behaviour of ants. In [3] presented a paper on the implementation of an already introduced Ant Colony Optimization Algorithm for Test Case Selection and Prioritization. This approach clearly explains the nature of ACO in identifying the possible paths and chooses the optimal solution from those paths. Results show that ACO leads to solutions that are in close proximity with optimal solutions.

ACO approach performs better than Genetic Algorithm as convergence is guaranteed, but time to convergence is uncertain. Moreover, in Non Deterministic polynomial time-hard (NP-hard) problems, high-quality solutions are required at a faster rate, but ACO focuses only on quality of solutions. Test case prioritization has also been done through PSO. Basic PSO is more appropriate to process static, simple optimization problems [9]. Moreover, it is hard to adapt to non-metric problem domains in PSO.

Bee Colony Optimization (BCO) is an emerging field for researchers. It has been applied to solve “Travelling Salesman Problem” which is a NP-Hard combinatorial problem where an optimal path is to be searched from source to destination.

Arvinder Kaur et al [10] presented the Bee Colony Optimization (BCO) algorithm for the fault coverage of a regression test suite. In the bee colony, Scout bees and Forager bees are accountable for the progress and maintenance of the colony. BCO algorithm developed for the fault coverage in a regression test suite makes use of the behaviour of these two bees. The BCO algorithm is designed to attain maximum fault coverage in minimal units of execution time of each test case.

The ABC algorithm is an optimization algorithm used to find an optimal solution to the problem in [3]. The algorithm works based on the honey bee foraging behaviour.

The main drawbacks of ABC are
• Slow convergence rate.
• As the random number is stochastic in basic ABC, certain good solutions are predictable to be skipped.

Due to the drawbacks of the above said optimization algorithms, an optimization algorithm which provides best convergence rate, less complexity, higher accuracy is required to solve the test case prioritization problem.

3. Methodology

In this approach, a combined form of PSO algorithm and ABC algorithm is used for optimizing the Test Cases.

In the proposed methodology each test case would symbolize a food source of the bees and the objective of this method is to find a best food source that refers to the test cases with maximum coverage.

The food source position of the bees corresponds to a potential solution of the optimization problem and the nectar amount match to the fitness of the associated solution.

3.1. ABC-PSO Hybrid Algorithm (PSABC)

ABC runs until its stopping condition to reach the maximum number of iterations. The end value of the
iteration is considered to be an optimal value of the individuals. The optimal values of individuals produced by the ABC algorithm are given as an input to the PSO algorithm. By this the PSO algorithm is initialized its position. PSO arbitrarily produces its initial individual sets, however in this situation of hybridization that is taken in consideration by giving the initiating point for the PSO whose final values for individuals is produced by the ABC. The advantages of proposed algorithm is

- Easy to implement.
- Broad applicability, even in complex functions, or with continuous, discrete or mixed variables.
- High flexibility, which allows adjustments and the introduction of specific knowledge of the problem by observing nature.
- Robust against initialization, regardless of feasibility and distribution of the initial solutions population.

3.2. Proposed Methodology

This research work proposes that the optimized test suite produced by the algorithm will comprise of all possible statements and faults in the program. ABC is functional to produce an Optimal Test suite by generating optimal test data which would have higher statement and path coverage. The test data will be the required input to be given to the SUT, for travelling along the path and vice versa. At first, the program is given to the Test Case optimization tool, which transforms the corresponding program into an equivalent Control Flow Graph (CFG). The independent paths from the start node to the end node are produced from CFG. Each independent path consists of number of normal nodes and predicate nodes. Every independent path would denote a Test Case. ABC algorithm is given to produce an Optimal Test suite through optimal test data which would cross the independent paths and then into to the test cases. The search bee would be a search agent which looks for the execution state of the SUT and also initiates the test cases with the initial test data through equivalence partitioning and boundary value analysis. Then the search agent computes the fitness value of each test node through evaluating the coverage of each node. This is repeated until an executable state of SUT is determined.

Then the search bee gives the fitness value of the traversed nodes/neighbouring nodes to the chosen agent [11]. The chosen bee evaluates the fitness value of traversed nodes and the neighbouring nodes. If the fitness value of the node obtained is greater than the neighbouring node’s fitness value, the node’s information is stored in the optimal test case repository. The node whose fitness value is observed less is discarded.

The algorithm for test case optimization using PSABC algorithm approach is seen below. In order to implement any algorithm, the algorithm must be converted into the pseudo code before programmatic development into an application.

The process is initiated with a collection of random particles (solutions), N. The ith particle is denoted by its position as a point in S-dimensional space, where S denotes the number of variables.

The following is the detailed algorithm

1. Initialize the test case performed by the search bee in the algorithm, and then evaluate the test cases.
2. Initialize the current traversal path, set cycle=1
3. Repeat the cycle
4. Generate the initial population and select the half part of bees as employed with PSO
5. Evaluate the fitness value for the initialized population
6. For each employee bee calculate the new test cases and find the fitness value for that new solution by applying greedy process.
7. Probability value for the new solution is calculated.
8. Above two processes is repeated for the onlooker bee, then replace it with the obtained new solution, which will be randomly produced and it is stored.
9. Add the test case to the optimal repository
10. In the next iteration scouts generate the new test data

ABC iterates till its stopping criterion is met which determines the maximum number of paths covered and faults covered. Then the optimal values of individuals obtained from the ABC are given to the PSO. Then the PSO is initiated. During the process, each particle observes three values namely its current position \( x_i \), the best position it arrived in previous cycles \( P_i \), its flying velocity \( V_i \). These three values are denoted as follows:

\[
\text{Current position } \quad x_i = (x_{i1}, x_{i2}, \ldots, x_{iS}) \\
\text{Best previous position } \quad P_i = (P_{i1}, P_{i2}, \ldots, P_{iS}) \\
\text{Flying velocity } \quad V_i = (V_{i1}, V_{i2}, \ldots, V_{iS})
\]

(1)

In each time interval (cycle), the position \( P_i \) of the best particle \( g \) is computed as the best fitness of all particles. Thus, each particle updates its velocity \( V_i \) to get closer to the best particle \( g \), as follows:

\[
\text{New } \quad V_i = \omega \times \text{current } V_i + c_1 \times \text{rand}() \times (P_i - X_i) + c_2 \times \text{Rand}() \times (P_i - X_i)
\]

(2)

As such, using the new velocity \( V_i \), the particle’s updated position becomes:

\[
\text{New position } \quad x_i = \text{current position } x_i + \text{New } V_i \text{ with } V_{\text{max}} \leq V_i \leq -V_{\text{max}}
\]

(3)

where \( c_1 \) and \( c_2 \) represent two positive constants named learning factors (usually \( c_1 = c_2 = 2 \)); \( \text{rand}() \) and \( \text{Rand}() \) denote two random functions in the range \([0,1]\); \( V_{\text{max}} \) is an upper limit on the maximum change of particle velocity [12], and \( \omega \) denotes an inertia weight employed as an enhancement proposed by Shi and Eberhart [13] to manage the influence of the previous history of velocities on the current velocity. The \( \omega \) balances the global search and the local search; and it is introduced to minimize linearly with time from a value of 1.4–0.5 [12]. For itself global search is initiated with a large weight and then decreases with time to favour local search over global search [14].

It is observed that the second term in equation (2) indicates cognition or the private judgment of the particle
when comparing its current position to its own best position. The third term in equation (2), denotes the social collaboration between the particles and compares a particle’s current position to that of the best particle [15]. Furthermore, in order to control the change of particles velocities, upper and lower bounds for velocity change is limited to a user-specified value of $v_{\text{max}}$. Once the new position of a particle is computed using equation (3), the particle, then, flies towards it [13]. Therefore, the main parameters used in the PSO are the population size (number of birds); number of generation cycles; the maximum change of a particle velocity $v_{\text{max}}$ and $\omega$. However in this situation of hybridization that is taken in consideration by giving the initiating point for the PSO whose final values for individuals is produced by the ABC.

The detailed pseudo code of the Test Case Optimization algorithm using PSABC approach is presented in the following section.

**Detailed pseudo-code of PSABC algorithm:**

1. **Initialize the test cases** which is performed by the search bee
   - Search for an executable state and evaluate the test nodes
   - Initialize the current traversal path as cycle=1
2. **Repeat**
   - Generate the initial population $X_{t1} = 1, 2, \ldots, N$
   - Select half part of bees as employed bee with PSO
   - Evaluate the fitness ($f_i, p_i$) of the population
   - Set cycle to 1
3. **Repeat**
   - For each employed bee Do
     - Produce new solution $V_i$
     - Calculate the value $f'_i$
     - Apply greedy selection process
     - Calculate the probability values $p_i$ for the solutions $X_i$
     - For each onlooker bee
       - Select a solution $X_i$ depending on $p_i$
       - Produce new solution $V_i$
       - Calculate the values $f'_i$ Apply greedy selection process
     - If there is an abandoned solution for the scout
       - Replace it with a new solution which will be randomly produced
     - Memorize the best solution so far
     - Add the test case to the optimal repository
     - cycle = cycle + 1
   - until cycle=MCN

To implement the above algorithm, the proposed approach uses the Test Suite Optimization tool to optimize the Test Cases by employing the PSABC algorithm. The tool considers a program as an input to generate independent paths. Using the generated independent paths Test Cases are traversed along the paths with the help of PSABC algorithm. By doing so, the test cases with maximum coverage (High fitness Value) are recognized. Finally the optimal Test Suite is generated as an output.

### 4. Simulation Results

The experiment is implemented in MATLAB. The test case prioritization technique’s basic evaluation is to have maximum number of faults covered and statement covered with minimum number of test cases required. In this approach, the execution time of every test case is also analyzed. The fault measuring technique used in fault coverage is based testing technique. In this example, there are test cases forming Test Suite ($T_S = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}\}$ and the faults covered by those test cases are represented as Faults Covered ($FC = \{F_1, F_2, F_3, F_4, F_5\}$). Similarly the statements covered by the test cases are denoted as Statements Covered ($SC = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$). The Control Flow Graph (CFG) is seen in figure 1.

![Fig. 1. CFG of the Quadratic Equation Code](image)

This section compares the performance of the proposed PSABC approach with the other optimization approaches such as ACO, PSO and ABC, in terms of percentage of statement coverage and fault coverage.

![Fig. 2. No. of Cycles Vs Statement Coverage (%) Comparison](image)
Table 1 and 2 clearly shows the Test cases with the faults and statements covered in particular execution time.

<table>
<thead>
<tr>
<th>Test Case/Faults</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>No. of Faults Covered</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>T2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>T4</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>T5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>T6</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>T7</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>T8</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>T9</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>T10</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case/ Faults</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>No. of Faults Covered</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>T2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>T4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>T5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>T6</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>T7</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>T8</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>T9</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>T10</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 2 shows the comparison of the number of cycles and the statement coverage in percentage. From the figure 2 that the proposed test case prioritization approach using PSABC provides better statement coverage when compared with ACO, ABC and PSO optimization approaches.

When the number of cycles increases, the Statement coverage also increases linearly. For instance, when the number of cycles is 14, the statement coverage of the ACO, PSO and ABC approach is 31%, 40% and 47% respectively. But, when the proposed PSABC approach is considered, the statement coverage attained is 51%. Similarly, for the other cycles, the proposed PSABC based Test case prioritization approach provides better results when compared with the other existing approaches.

Figure 3 shows the fault coverage comparison in percentage for the approaches such as ACO, PSO, ABC and PSABC. It can be observed from the graph that, there is significant increase in the faults coverage with the increase in the number of cycles. The proposed approach outperforms the other two approaches in terms of the fault coverage. For example, considering the number of cycles is 14, the fault coverage obtained by the approaches like ACO, PSO and ABC are 42%, 54% and 59% respectively. But, when the proposed PSABC approach is considered, the fault coverage obtained is 65% which is higher than the approaches taken for consideration.

Thus, it can be observed from the simulation results that the test cases are prioritized based on higher statement coverage and fault coverage using the PSABC approach.

![Fig. 2. No. of Cycles Vs Statement Coverage (%) Comparison](image)

5. Conclusion

Testing ensures that the software meets the user conditions and necessities. Effectual generation of test cases and prioritization of test cases has to be addressed in the field of Software Testing. Factors like effort, time and cost of the testing are factors influencing these as well. A
number of research work have been proposed in the literature for test case prioritization. The main aim for prioritization of test cases is to minimize the cost and time of regression testing. The objectives considered in this research work are statement coverage and fault coverage within a minimum execution time. This research work aims in attaining test case prioritization results using PSABC. It is observed from the experimental results that the proposed PSABC based test case prioritization based approach provides better results when compared with ACO, PSO and ABC.

References