An Efficient Specific Update Search Domain based Glowworm Swarm Optimization for Test Case Prioritization

Beena Raman and Sarala Subramani
[1]Department of Information Technology, Bharathiar University, India

Abstract: *Software testing is an important activity that is carried out during the software development life cycle. Regression testing means re-executing test cases from existing test suites to assure that the modifications done to the existing software have no adverse effects. During regression testing, new test cases are not created but previously created test cases are re-executed. The ideal regression testing is to rerun all the test cases, but due to time and cost constraints only a subset of test cases are rerun based on regression testing techniques. The various regression testing techniques are test case minimization, test case selection and test case prioritization. In this paper, an approach to solve test case prioritization based on efficient swarm intelligence approach called Glowworm Swarm Optimization (GSO) is proposed. This research work focuses on a conception of definite updating search field at glowworm updating position stage. Based on the Specific Update search domain based GSO (SU-GSO) approach, an optimal number of test cases to be executed on Software Under Test (SUT) is obtained. The objectives of this research work are to maximize the path coverage and fault coverage for getting the optimal prioritized test cases. The resulting solution guarantees an optimal ordering of test cases and the performance of the proposed SU-GSO is compared with other optimization techniques such as Particle Swarm Optimization (PSO) and artificial Bee Colony Optimization (BCO).*

Keywords: *Regression testing, test case prioritization, GSO.*

Received May 20, 2013; accepted August 10, 2014; published online August 9, 2015

1. Introduction

Software bugs are almost present in most of the software modules which are being developed by the software developers as the complexity of software is usually intractable and software developers have only restricted potential to handle complexity. Identifying the design defects in software is very tough due to complexity. Since, software is not continuous, testing boundary values are inadequate method to assure accuracy. Thus, the entire values are required to be evaluated and confirmed, but this entire testing is not feasible [28].

The software maintenance phase requires efficient regression testing process. It is necessary to retest the existing test suite whenever any alterations are done to the software. Software testing is an important part of the software development process. It is one of the major and primary techniques for achieving high quality software. It is the process of evaluating a system or its components with the intent to find that whether it satisfies the specified requirements or not. Regression testing is the phenomenon of re-running the test cases from the test suite to assure error free modified software. It guarantees that modifications in the software have not influenced functional characteristics of software [1].

But, software developers often have time and budget limits in running all the test cases within the particular constraints. Thus, this approach is quite expensive and time consuming. Therefore, in order to deal with the problem of time and budget constraint, test case minimization, selection and prioritization techniques [30] have been used for regression testing for effective cost reduction in regression testing.

The proposed research work mainly focuses on the test case prioritization. In test case prioritization, the test cases are ordered priority-wise based on the objectives rate of fault detection and average statement coverage, such that highest priority test case is to be executed first and so on. Prioritization can offer earlier feedback to testers and management and facilitate software developers to begin debugging much earlier. It can also raise the probability that if testing ends in advance, only necessary and worthy test cases have been rerun [20, 21].

Test case prioritization looks for identifying potential ordering of test case execution for regression testing. The efficiency of the test case prioritization lies in revealing the faults at the earliest [31]. But, the nature and locality of actual faults are usually not known in advance and thus, test case prioritization approach have to largely depend on available substitutes for prioritization criteria. Structural coverage, requirement priority and mutation score have been used in the literature as criteria for performing prioritization [5, 24]. But, there is no single prioritization criterion whose results dominate the others.

Several numbers of test cases will be formed through exhaustive testing [29]. But, all the test cases cannot be considered and only a few test cases will be performing well if implemented in testing the software. This is the main issue considered in this paper. This problem necessitates obtaining the result based on the statement coverage and fault coverage in particular time.

The test cases are prioritized based on the average statement coverage and rate of fault detection. The prioritization will be based on certain test adequacy criteria or fitness value [9]. Furthermore, through polynomial bounded computation, most of the complex multivariable optimization problems [4] cannot be solved accurately. This formulation induces to develop a novel approach through search based intelligent selection and prioritization of test case.

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

Thus, this approach uses swarm intelligence based technique for test case prioritization. A number of swarm intelligence approaches have been observed to produce significant results in terms of its accuracy, convergence behavior, time taken etc., this research uses a recent and efficient swarm intelligence approach called as the Glowworm Swarm Optimization (GSO) for getting the optimal test case prioritization results [15, 17]. GSO is used to produce the optimal test case prioritization which is adequate to cover the statements and faults in the software. This research work focuses on the multi objective criteria namely the average statement coverage and rate of fault detection.

2. Literature Survey

This section presents an overview of the related work of test case prioritization using some of the search based approaches and met heuristics. The regression testing has been solved using optimization approaches like Genetic Algorithm (GA) [12], Ant Colony Optimization (ACO) [23], etc.

A cost cognizant test case prioritization approach is presented by [8] with the application of previous data and GA. A controlled experiment has been carried out to evaluate the performance of the approach. But, the main drawback of this approach is that it does not consider the test cases similarity. A novel technique called Edge Partitions Dominator Graph-Genetic Algorithm (EPDG-GA) which uses the EPDG and GA for branch coverage testing is proposed by [3].

The main drawback of the GA is its slow convergence. Moreover, it is more complex since, it lacks rank based fitness function which reduces complexity.

ACO is a bio inspired approach based on the real life behavior of ants. In [26], a technique on application of ACO algorithm for test case selection and prioritization is proposed. This work clearly explains the graphical depiction of food search of the ants which results in finding different paths and choosing the optimal path. Results indicate that ACO results in solutions that are in close proximity with optimal solutions.

ACO approach performs better than GA as convergence is guaranteed, but the time period for convergence is uncertain. Furthermore, in NP-hard problems, high-quality solutions are needed at a faster rate, but ACO concentrates only on quality of solutions.

More recently, Bee Colony Optimization (BCO) has emerged as a potential technique in the field of swarm intelligence. BCO has been applied in various optimization problems like "Travelling Salesman Problem" [22] which is a NP-Hard combinatorial issue in which an optimal path is to be found from source to destination for a travelling salesman.

Kaur and Goyal [11] an approach for the fault coverage regression test suite prioritization based on the BCO algorithm is presented. The most difficult task in regression testing is the size of the regression test suite and its selection process due to its time and budget constraints. In BC, Scout bees and forager bee are accountable for the development and maintenance of the colony. Based on the nature of these two bees, BCO algorithm for the fault coverage regression test suite has been developed. BCO algorithm has been designed for fault coverage to obtain maximum fault coverage in minimal execution time of each test case. But, the main drawbacks of BCO are:

- Slow convergence rate.
- As the random number is stochastic in basic BCO, certain good solutions are predictable to be skipped.

The application of Hybrid Particle Swarm Optimization (HPSO) algorithm in regression technique is proposed by Kaur and Bhatt [10]. The criterion considered is maximum fault coverage in minimum execution time. HPSO is an integration of Particle Swarm Optimization (PSO) technique and GA to enhance the search space for the solution. GA offers optimized approach to carry out prioritization in regression testing and on integrating it with PSO technique provides fast solution. GA has been used is Mutation operator which facilitates the search engine to assess all features of the search space.

A large number of fundamental variations have been formulated to enhance the speed of convergence and significance of solution found by PSO. But, fundamental PSO is more suitable to process static, simple optimization problem. Furthermore, it is very difficult to deal with non-metric problem domains in PSO [7].

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

Thus, in this research work, GSO is used to solve the test case prioritization problem. GSO is a novel technique of swarm intelligence based algorithm for optimizing multiple functions raised by Krishnanad and Ghose [16]. This algorithm has become one of the active research areas of swarm intelligence. Because of its potential influence, it's been used at noisy text of sensor and simulating robots [13]. Hence, this research work uses GSO to attain the optimal results.

Li *et al*. [18] gives empirical study results of two met heuristic search techniques and three greedy search techniques applied to six programs for regression test case prioritization.

Test case prioritization approach was first proposed by Wong *et al*. [27], which is aimed at ranking the test cases in an optimal ordering. Even if the testing is terminated at a point, the maximum benefit can be obtained.

Srivastava [25] Praveen initiated a novel test case prioritization algorithm that calculates average faults observed per minute.

A regression testing technique for test case prioritization based on code coverage criteria is recommended by Aggarwal *et al*. [2].

3. Methodology

This research work uses an efficient GSO algorithm for the formulation of test case prioritization. In this proposed approach, each test case would denote a glow source and the aim of the approach would be to determine best sources i.e., test cases with maximum statement coverage and maximum fault coverage.

The main aim of the proposed approach is to determine the optimal number of test cases with higher statement coverage and fault coverage. In order to, handle this issue, GSO approach gather the test cases and then calibrate the local decision range function which is in-turn used to identify the optimal test cases with maximum statement coverage and fault coverage.

On choosing the population randomly from a given issue, local-decision range is assigned based on the position of the GSO algorithm. This local-decision range represents to a possible solution of the optimization problem and the luciferin decay constant corresponds to the quality (fitness) of the associated solution [14]. In this process of test case prioritization, the position of glowworm (luciferin update) is considered as number of statements and faults covered by that glowworm. The regression testing mainly focuses on total statement and fault covered in less time. The stopping criterion is to be decided, on the basis of which glowworm optimization algorithm will end.

This research work proposes that the optimized test suite produced by the algorithm will comprises of all possible statements and faults in the program. The program is given to the test case optimization tool. GSO is applied to produce an optimal test suite by generating optimal test data which would have higher statement and path coverage. The glowworm distributes in the objective function definite space. In GSO, brightness is regarded as search agent for the execution state of the Software Under Test (SUT) and also initializes the test cases by defining the parameter with the initial test data with the aid of corresponding partitioning and boundary value analysis. The search agent computes the local decision range based on the brightness of each test node by estimating the statement and fault coverage. This process is repeated until an executable state of SUT is determined. The glowworm optimization can be distributed in the objective function definite space. Their brightness is based on the position of objective function value. With the brighter glow, the better is the position which in turn results in good target value. The glow looks for the neighbor group in the local-decision range and a brighter glow in the set has a higher pull to attract this glow towards this traverse and the flight direction each time will change along with the choice of neighbor. In the mean time, local-decision range size will be based on the neighbor quantity which means when the neighbor density is low, glow's policy-making radius will expand which in turn looks for more neighbors, on the other hand, the policy-making radius minimizes if the neighbor density is high. Ultimately, a number of glowworm returns collects at the multiple optima of the given objective function. If the condition is not met, then the new set of test data is formed from the abandoned repository and again the same process is repeated.

Though GSO is much suited for various applications, yet the search accuracy and post iteration is not high. This research work uses an idea of definite updating search field at glowworm updating position stage for improving the performance of the GSO. The algorithm for test case optimization using Improved GSO approach is presented in the next section.

3.1. Specific Update Search Domains of Glowworm Swarm Optimization

A novel algorithm called Specific Update search domains of Glowworm Swarm Optimization (SU-GSO) is used in this research work for test case prioritization. SU-GSO makes the position update of glowworm which makes it get to the optimal solution which in turn increases the accuracy and speeding up convergence [19]. Each glowworm i encodes the object function values $R_1(x_i(t))$ and $R_2(x_i(t))$ at its current location $x_i(t)$ into a luciferin value l_i and transmits the data within its neighborhood. The multiple objective functions considered in this research work are Statement Coverage (SC) which corresponds to R_1 is given in Equation 1 and Fault Coverage (FC) which corresponds to R_2 is given in Equation 2. It means that, this work concentrates on maximizing the statement coverage and fault coverage for a given time.

$$R_1(SC) = \text{Average Statement Coverage} \\ = \text{No. of Statements covered by Total no. of Statement} \tag{1}$$

$$R_2(FC) = \text{Rate of Fault Detection} \\ = \text{No. of Faults covered by Execution Time} \tag{2}$$

$$\text{Objective Function } R = \alpha * R_1 + \beta * R_2$$

Where α and β are weights, such that $\alpha+\beta=1$.

The set of neighbors $V_i(t)$ of glowworm i comprises of those glowworms that have a fairly better luciferin value and that are positioned within a dynamic decision domain, and updating at each iteration. Local-decision range update is given in Equation 3.

$$r_d^i(t+1) = min\left\{r_s, max\left\{0, r_d^i(t) + \beta(n_t - |V(t)|)\right\}\right\} \tag{3}$$

Where $r_d^i(t+1)$ denotes the glowworm i's local-decision range at the $t+1$ iteration, r_s denotes the sensor range, n_i represents the neighborhood threshold, the parameter β affects the rate of change of the neighborhood range. The number of glow in local-decision range is given in Equation 4.

$$N_i(t) = \left\{R : \left\|x_j(t) - x_i(t)\right\| < r_d^i ; l_i(t) < l_j(t)\right\} \tag{4}$$

Where $X_j(t)$ denotes the glowworm i'*s* position at the t iteration, $l_j(t)$ represents the glowworm i'*s* luciferin at the t iteration and the set of neighbors of glowworm i comprises of those glowworms that have a relatively higher luciferin value and that are positioned within a dynamic decision domain whose range r_d^i is bounded above by a circular sensor range $r_s(0 < r_d^i < i)$. Each glowworm i chooses a neighbor j with a probability $p_{ij}(t)$ and moves toward it. These movements that depend only on local data facilitate the glowworms to separate into disjoint subgroups and demonstrate a simultaneous behavior toward and eventually co-locate at the multiple optima of the given objective function probability distribution given in Equation 5 is used to select a neighbor.

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\Sigma k \in N_i(t)\, l_k(t) - l_i(t)} \tag{5}$$

Movement update is given in Equation 6:

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\left\|x_j(t) - x_i(t)\right\|}\right) \tag{6}$$

Luciferin-update is given in Equation 7:

$$l_i(t) = (1-\rho)\, l_i(t-1) + \gamma R(x_i(t)) \tag{7}$$

Where $l_j(t)$ depends on luciferin value of glowworm i at the t iteration, ρ ™(0, 1) results in the reflection of the cumulative goodness of the path followed by the glowworms in their current luciferin values, the parameter γ only scales the function fitness values, $R_1(x_i(t))$ is the value of test function.

In fundamental GSO algorithm, a conception of specific update at glowworm position to organize alteration of the glowworm position. Because of this approach, after the specific update of the glow position always in present most superior individual margin to improves the convergence rate of the algorithm. Improvement algorithm according to update location is given in Equation 8.

$$x_i(t) = x_{best}(t) + (rand - 0.5) \tag{8}$$

Where $x_{best}(t)$ represents the best one at the t iteration, $x_i(t)$ denotes the position after updated of glow that need to change.

3.2. SU-GSO Algorithm

The proposed algorithm is formulated as follows:

- *Step* 1: Initialize the test cases by defining the parameters ρ, γ, β, s, l_0, m, n with values 0.4, 0.6, 0.08, 0.03, 5,5 respectively.
- *Step* 2: For each test cases calculate the luciferin value and update it according to Equation 7.
- *Step* 3: Find the set of neighbors of each test case that have a relatively higher luciferin value according to Equation 4.
- *Step* 4: If the test case does not satisfy Equation 4, then renew its position with Equation 8.
- *Step* 5: Select the distribution using Equation 5 and update it with Equation 6.
- *Step* 6: Enhance the search radius according to Equation 3.
- *Step* 7: Check whether the condition is satisfied.
- *Step* 8: Proceed for the next iteration and wait for the termination condition.

3.3. SU-GSO Flow Chart

Figure 1 gives the flow chart of the proposed SU-GSO algorithm's flow chart.

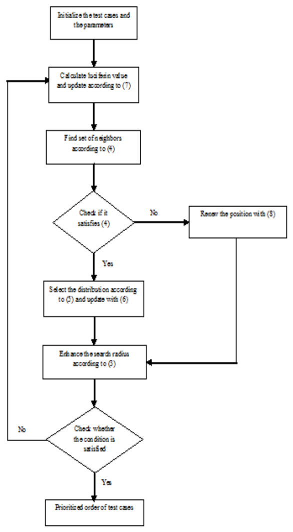
Initialize the test cases and the parameters
Calculate luciferin value and update according to (7)
Find set of neighbors according to (4)
Check if it satisfies (4)
No
Renew the position with (8)
Yes
Select the distribution according to (5) and update with (6)
Enhance the search radius according to (3)
No
Check whether the condition is satisfied
Yes
Prioritized order of test cases

Figure 1. Proposed SU-GSO algorithm flow chart.

4. Experiments and Results

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 is fault coverage based testing technique. In this example, there are test cases forming *Test Suite* (*TS*)={*T*1, *T*2, *T*3, *T*4, *T*5, *T*6, *T*7, *T*8} and the faults covered by those test cases are represented as *Faults Covered* (*FC*)={*F*1, *F*2, *F*3, *F*4, *F*5, *F*6}. Similarly the statements covered by the test cases are denoted as *Statements Covered* (*SC*)={*S*1, *S*2, *S*3, *S*4, *S*5}. Tables 1 and 2 show the Test cases with the faults and statements covered in particular execution time.

Table 1. Test cases with fault coverage and execution time.

Test Cases/ Faults	F1	F2	F3	F4	F5	F6	No. of Faults Covered	Execution Time
T1	x		x	x			3	10
T2		x	x		x	x	4	8
T3	x			x			2	11
T4			x		x	x	3	15
T5	x	x		x	x		4	11
T6	x					x	2	9
T7		x	x	x	x		4	8
T8	x					x	2	6

Table 2. Test cases with statement coverage.

Test Case/Faults	S1	S2	S3	S4	S5	No. of Statements Covered
T1		x		x		2
T2	x		x		x	3
T3	x	x		x		3
T4		x	x			2
T5		x	x	x	x	4
T6	x					1
T7	x	x			x	3
T8	x			x		2

The performance of the proposed SU-GSO approach is compared with the other optimization approaches such as PSO and Artificial Bee Colony (ABC) in terms of percentage of statement coverage, fault coverage and both. It is clearly observed from the Figure 2 that the proposed test case prioritization approach using SU-GSO provides better statement coverage when compared with ABC and PSO optimization approaches.

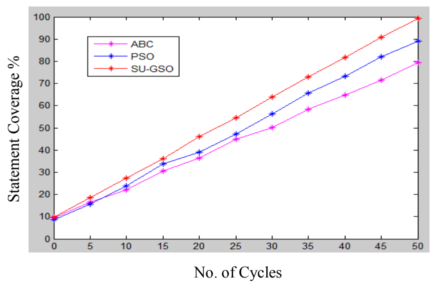

Figure 2. No. of cycles vs. statement coverage (%) comparison.

Figure 3 shows the fault coverage comparison in percentage for the approaches such as PSO, ABC and SU-GSO. The proposed approach outperforms the other two approaches in terms of the fault coverage.

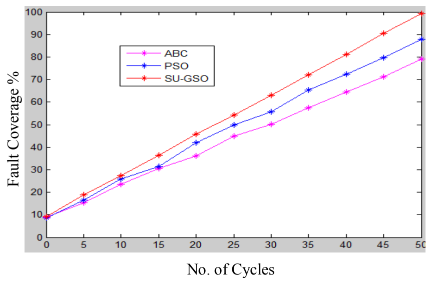

Figure 3. No. of cycles vs. fault coverage (%) comparison.

It can be observed from the graphical representation that the test cases are prioritized based on higher statement coverage and fault coverage are selected as the optimal test cases.

Figure 4 shows the graphical representation of the objective function R Vs the number of cycles. The proposed SU-GSO is observed to produce better results than the other two optimization algorithms taken for comparison.

For instance, when the number of cycles is 45, the coverage obtained by the ABC approach is 70% and the coverage obtained from PSO approach is 78%, whereas the proposed SU-GSO approach attains coverage of about 90%. Thus, the proposed SU-GSO approach outperforms the other two approaches.

This would result in the prioritization of the test cases in a suitable and appropriate order which in turn improves the overall performance of the system.

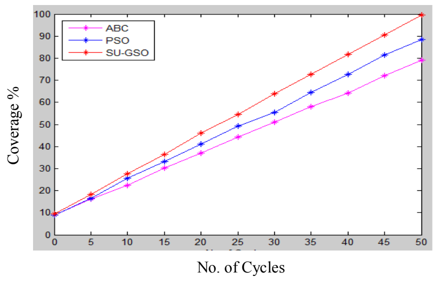

Figure 4. No. of cycles vs. coverage (%) (objective function R)

4.1. APFD Metric

This performance of the proposed SU-GSO based test case prioritization has been represented through APFD representation. The APFD percentage is calculated by concerning test suite selected from above program solution [6].

To quantify the goal of increasing a subset of the test suite's rate of fault detection, APFD metric developed by Elbaum *et al*. [6] is used to measures the average rate of fault detection per percentage of test suite execution. The APFD is calculated by taking the weighted average of the number of faults detected during the run of the test suite. APFD can be calculated using a notation:

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots TF_m}{nm} + \frac{1}{2n} \tag{10}$$

Where *T*: The test suite under evaluation, *m*: The number of faults contained in the program under test *P*, *n*: The total number of test cases, TF_i: The position of the first test in *T* that exposes fault *i*.

So, as the formula for APFD shows that calculating APFD is only possible when prior knowledge of faults is available. APFD calculations therefore are used only for evaluation.

5. Conclusions

Test case prioritization has become an active area of research in the field of software testing. A number of research works 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. This research work focuses on novel test case prioritization considering multi objective criteria and an efficient optimization technique. The objectives considered in this research work are average statement coverage and rate of fault detection. This research work succeeds in attaining efficient test case prioritization results using SU-GSO. The performance of the approach is compared with other optimization approaches such as PSO and ABC. It is observed from the experimental results that the proposed SU-GSO based test case prioritization based approach provides better results when compared with PSO and ABC.

References

[1] Agrawal H., Horgan R., Krauser W., and London S., "Incremental Regression Testing," *in Proceedings of International Conference on Software Maintenance*, CSM-93, Montreal, Canada, pp. 348-357, 1993.

[2] Aggrawal K., Singh Y., and Kaur A., "Code Coverage based Technique for Prioritizing Test Cases for Regression Testing," *ACM SIGSOFT Software Engineering Notes,* vol. 29, no. 5, pp. 1-4, 2004.

[3] Chen C., Xu X., Chen Y., Li X., and Guo D., "A New Method of Test Data Generation for Branch Coverage in Software Testing based on EPDG and Genetic Algorithm," *in Proceedings of the 3rd International Conference on Anti-Counterfeiting, Security, and Identification in Communication*, Hong Kong, pp. 307-310, 2009.

[4] Deb K., "Multi-Objective Optimization using Evolutionary Algorithms," John Wiley & Sons, 2001.

[5] Do H. and Rothermel G., "On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 733-752, 2006.

[6] Elbaum S., Malishevsky G., and Rothermel G., "Test Case Prioritization: A Family of Empirical Studies," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 159-182, 2002.

[7] Farzi S., Shavazi R., Pandari A., and Graduated A., "Using Quantum-Behaved Particle Swarm Optimization for Portfolio Selection Problem," *the International Arab Journal of Information Technology*, vol.10, no. 2, pp. 111-119, 2013.

[8] Huang C., Huang Y., Chang R., and Chen Y., "Design and Analysis of Cost-Cognizant Test Case Prioritization Using Genetic Algorithm with Test History," *in Proceedings of the 34th Annual Computer Software and Applications Conference*, Seoul, South Korea, pp. 413-418, 2010.

[9] Karaboga D. and Basturk B., "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony Algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459-471, 2007.

[10] Kaur A. and Bhatt D., "Hybrid Particle Swarm Optimization for Regression Testing," *International Journal on Computer Science and Engineering*, vol. 3, no. 5, pp.1815-1824, 2011.

[11] Kaur A. and Goyal S., "A Bee Colony Optimization Algorithm for Code Coverage Test Suite Prioritization," *International Journal of Engineering Science and Technology*, vol. 3, no. 4, pp. 2786-2795, 2011.

[12] Kaur A. and Goyal S., "A Genetic Algorithm for Regression Test Case Prioritization using Code Coverage," *International Journal on Computer Science and Engineering*, vol. 3, no. 5, pp. 1839-1847, 2011.

[13] Krishnanand N. and Ghose D., "A Glowworm Swarm Optimization based Multi-Robot System for Signal Source Localization," *Design and Control of Intelligent Robotic Systems*, vol. 1, no. 1, pp. 49-68, 2009.

[14] Krishnanand N. and Ghose D., "Chasing Multiple Mobile Signal Sources: A Glowworm Swarm Optimization Approach," *in Proceedings of the 3rd Indian International Conference on Artificial Intelligence*, 2007.

[15] Krishnanand N. and Ghose D., "Glowworm Swarm Optimisation: A New Method for Optimising Multi-Modal Functions," *International Journal of Computational Intelligence Studies*, vol. 1, no. 1, pp. 93-119, 2009.

[16] Krishnanand N. and Ghose D., "Theoretical Foundations for Rendezvous of Glowworm-Inspired Agent Swarms at Multiple Locations," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 549-569, 2008.

[17] Krishnanand N., "Glowworm Swarm Optimization: A Multimodal Function Optimization Paradigm with Applications to Multiple Signal Source Localization Tasks," *International Journal of Computational Intelligence Studies*, vol. 1, no. 1, pp .93 2009.

[18] Li Z., Harman M., and Hierons M., "Search Algorithms for Regression Test Case Prioritization," *IEEE Transaction on Software Engineering*, vol. 33, no. 4, pp. 225-237, 2007.

[19] Liu J., Zhou Y., Huang K., Ouyang Z., and Wang Y., "A Glowworm Swarm Optimization Algorithm Based on Definite Updating Search Domains," *Journal of Computational Information Systems*, vol. 7, no. 10, pp. 3698-3705, 2011.

[20] Rothermel G., Untch H., Chu C., and Harrold J., "Prioritizing Test Cases for Regression Testing," *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 929-948, 2001.

[21] Rothermel G., Untch H., Chu C., and Harrold J., "Test Case Prioritization: An Empirical Study," *in Proceedings of International Conference on Software Maintenance*, Oxford, UK, pp. 179-188, IEEE, 1999.

[22] Saab M., El-Omari T., and Owaied H., "Developing Optimization Algorithm using Artificial Bee Colony System," *Ubiquitous Computing and Communication Journal*, vol. 4, no. 3, pp. 391-396, 2009.

[23] Singh Y., Kaur A., and Suri B., "Test Case Prioritization using Ant Colony Optimization," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 4, pp.1-7, 2010.

[24] Srikanth H., Williams L., and Osborne J., "System Test Case Prioritization of New and Regression Test Cases," *in Proceedings of International Symposium on Empirical Software Engineering*, pp. 1-10, 2005.

[25] Srivastava R., "Test Case Prioritization," *Journal of Theoretical and Applied Information Technology*, vol. 4, no. 3, pp. 178-181, 2008.

[26] Suri B. and Singhal S., "Implementing Ant Colony Optimization for Test Case Selection and Prioritization," *International Journal on Computer Science and Engineering*, vol. 3, no. 5, pp. 1924-1932, 2011.

[27] Wong E., Horgan R., London S., and Mathur P., "Effect of Test Set Minimization on Fault Detection Effectiveness," *Software Practice and Experience*, vol. 28, no. 4, pp. 347-369, 1998.

[28] Yang K. and Chao A., "Reliability-Estimation and Stopping-Rules for Software Testing, based on Repeated Appearances of Bugs," *IEEE Transactions on Reliability*, vol. 44, no. 2, pp. 315-321, 1995.

[29] Yoo S. and Harman M., "Pareto Efficient Multi-Objective Test Case Selection," *in Proceedings of International Symposium on Software Testing and Analysis*, London, UK, pp.140-150, 2007.

[30] Yoo S. and Harman M., "Regression Testing Minimization, Selection and Prioritization: A Survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67-120, 2012.

[31] Yoo S., Harman M., Tonella P., and Susi A., "Clustering Test Cases to Achieve Effective and Scalable Prioritizations Incorporating Expert Knowledge," *in Proceedings of the 18th International Symposium on Software Testing and Analysis*, Chicago, USA, pp. 201-212, 2009.

Beena Raman is a Doctoral Research student in the department of Information Technology, Bharathiar University, Coimbatore. She has published many papers in International Journals and Conferences. Her areas of interest include software testing, object oriented programming and compiler design.

Sarala Subramani received her PhD degree from Anna University, Chennai. Currently, she is an Associate Professor in Department of Information Technology, Bharathiar University, Coimbatore. She has a teaching and research experience of 10 years. She has published papers in various International journals and conferences. Her areas of interest include software testing, software engineering, object oriented programming concepts, data structures and compiler design.