Indian Journal of Science and Technology, Vol 9(40), DOI: 10.17485/ijst/2016/v9i40/99314, October 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Gravitational Bee Search Algorithm with Fuzzy Logic for Effective Test Suite Minimization and Prioritization

J. Srividhya* and R. Gunasundari

Department of Computer Science, Karpagam University, Coimbatore - 641021, Tamil Nadu, India;
jsrividhya.ku2011@gmail.com, gunasoundar04@gmail.com

Abstract

Objectives: Software testing is the significant part of software development and is essential to confirm the quality of the software. The test suites developed for this purpose can be used again and updated repeatedly as the software advances. Subsequently, novel test cases will be added to the test suite and because of that, the size of the test suite will become bigger. Moreover, the test suite becomes redundant. Thus executing/re-executing the large test suite consumes more time and also increases the cost of testing. Therefore, in order to minimize the cost and the time of testing, it is essential to minimize the test suite. **Methods/Statistical Analysis:** Thus, the focus of this paper is to minimize the test suite by discovering a group of test cases that gives the same or better coverage as the original test suite based on some condition. **Finding:** In this study, the minimization is achieved by using a Gravitational Bee Search (GBS) algorithm, this algorithm is derived by combining artificial bee colony and gravitational search algorithms. Then, the Fuzzy operation is applied for prioritization to achieve efficient test suite. The algorithm searches for the optimum solution by calculating fitness values using coverage information. The search process is repetitive until a reduced test suite is identified. **Application/ Improvement:** The proposed algorithm is applied on an online ticket booking system and the results shows that the proposed system is approximately 15% to 20% more efficient than the existing system respective to the number of test cases and execution time.

Keywords: Gravitational Bee Search, Software Testing, Test Suite Minimization, Test Coverage

1. Introduction

As software systems become more complex in terms of proving their exactness, it also becomes one of the challenging jobs. For instance, in greatly configurable systems, there are different types of existing optional features which may be used for involuntary interaction with each other code[1]. In verification methods like model based testing, formal verification may need expensive resources[2]. Because of their size and software complexity, deciding what to test from a possible test space, what kind of testing and test cases needs to be included are difficult. Selecting the right set of tests to minimize the risk of defects and redundancy is the key phase of planning software testing.

Before testing a code of software (program), testers have to form the objective of the testing process. The objective of testing is taken into account as a defined set of requirements. The test cases should be formed to mutually satisfy the requirements. A set of test cases can satisfy all testing needs is called as a test suite[3]. The development of test suite is an expensive procedure. Even though it is thoroughly maintained, the test suite can rise in size. Executing the entire test suite is not always conceivable, as it consumes lot of time to execute all the test cases in the test suite.

Many researchers have given different types of methods to minimize the test suite. These methods are focusing on reducing the testing cost by using a subset of test suite

*Author for correspondence

that can provide equally efficient result as original test suite.

The previous test suite minimizing methods considerably minimized the test suite's size. But they also needs to consider how well these minimized test suites can are associated with their original test suites[4]. The purpose of executing test suite is to identify in the given software. So the test suit quality is measured by single evaluation with respect to fault detection[5]. The exchange between time to execute and manage the test suite, and their fault detection effectiveness should be considered when employing the test suite minimizing methods.

In[6], the author investigates the utilization of Particle Swarm Optimization (PSO) for test suite minimization issues. In this work, a new criterion is used, which is a combination of a test execution cost and coverage to take a decision about reducing the test suite size. Then the evaluation results were examined by associating the results from Genetic Algorithm test suite minimization method. The assessment shows that the proposed PSO based test suite minimization process provides better results on cost and size reduction.

In[7], the author introduced a nature inspired algorithm, for example, PSO, GA, BFO to find an optimal test suite for multi-objective test suite reduction problem. The multiple objectives are fault history, cost, and code coverage. The regression test suite optimization is one of the efficient methods to lessen the cost and time of the testing process. In this effort, the author presents different types of nature inspired algorithms to overcome the traditional issues at the time of test suite generation.

In[8], the author presents a combined ABC integrated with Fuzzy C-Means (FCM) and PSO for generating an optimal test-suite. Fuzzy logic works with reasoning that is inexact rather than static and exact[9]. The ABC algorithm contains three different phases as scout, employed and onlooker bee phase, onlooker bee phase. Initially, the ABC worker identifies the nodes with the greatest coverage. Then the PSO algorithm is used to find the highest usage. Based on the hybrid optimization ABC and PSO algorithm, the optimum test suite is achieved by running the given raw test suite repeatedly. The evaluation result shows that the final test suite has less complexity with improved quality in terms of test suite generation.

In[10], the author presents an efficient Glowworm Swarm Optimization (GSO) method to resolve the test suite prioritization challenge. This exertion mainly focuses on updating the search field by glowworm at the stage of position updating. Based on this proposed method, the Software under Test is used to obtain the optimal test cases. The main objective of this proposed work is to upturn the fault coverage and path coverage. This method gives the promising results on obtaining an optimal test suite.

In[11], the author proposes two Multi-Objective Optimization (MOO) algorithms such as Binary Multi-Objective Particle Swarm Optimization with Crowding Distance and Roullete Wheel (BMOPSO-CDR) and BMOPSO-CDR with the Harmony Search (BMOPSO-CDRHS). The proposed algorithm solves the search based optimization strategy issues. The experiment covers both the functional and structural testing scenarios. The result shows the better results of BMOPSO-CDRHS, in terms of search based optimization strategies, when compared with the Multi-Objective Binary Harmony Search (MBHS) algorithm and Non-dominated Sorting Genetic Algorithm (NSGA-II).

In[12], the author formalizes the criteria as a combination of three hybrid SI algorithms. Three main metrics such as Choice, Merge, and Rank are considered for the test suite minimization process. In this work, recast is applied to evaluate and create new hybrid criteria from previous research. The experimental results show better performance in terms of Formulation, Rank and Merge.

2. Problem Statement

The test suites are significant to software system. Choosing a subset from the current test suite for the given requirements is the main process. Thus, the minimization of test suite problem is explained as follows.

A set of test cases T is defined as $\{t_1, t_2, t_3, \ldots t_n\}$. The set of test requirements which must be covered is defined as $\{r_1, r_2, r_3, \ldots r_m\}$. The focus of the test suite minimization process is to find a sub group of T and it must be in minimal cardinality that trains the same set of requirement, those trained by the un-reduced test suite T .

Let us consider $F(T_j)$ be a function that yields the requirements to the subset. The individual test case T_j should cover the above mentioned requirements. The coverage of test case t is

$$Coverage\ (t) = 100X\frac{\left|U_{t_j=1}\{F(T_j)\}\right|}{k} \quad (1)$$

where $U_{t_j=1}\{F(T_j)\}$ is the functional requirement's union subsets. The nominated test cases T_j for which $t_j = 1$ covers the mentioned subsets.

The 'to be minimized function' (execution cost) denotes the time taken to execute the nominated test suite. Each and every test case $T_j \in T$ has a cost value c_j . The overall cost of a solution is defined as t is as below

$$Cost\ (t) = \sum_{t_j=1} c_j \quad (2)$$

The proposed Gravitational Bee Search (GBS) algorithmis provides a better test suite with respect to the functions $Cost\ and\ Coverage$.

3. Gravitational Search Algorithm (GSA)

GSA is a well-known, newly emerged population based stochastic optimization approach which works by combining the processes of mass interaction and gravity. This iterative approach does a multi-dimensional search and mimics the mass interactions, which are done under the inspiration of gravitation in physics[13]. In GSA, the particles and their enactments are assessed by their masses. In this process, each and every particle is defined as an applicant solution to the given search problem. All particles interests each other by their gravity power, and this power can cause a universal movement of particles with greater masses. Therefore, masses work together with an interaction by gravitational force. The particle with higher mass has higher fitness values, which signifies the good optimal resolution to the given test suite searching issue. This method proves that the proposed algorithm yields convergent performance and global solution.

The collection of bodies, that means the search agents are used to discover the finest solution. After formatting the search space, rules must be specified to govern the collection of bodies.

4. Artificial Bee Colony Algorithm (ABC)

ABC algorithm is one of the eminent swarm based Meta heuristic algorithm which simulates the real honey bee's searching behavior. ABC is known as an optimization tool which gives a population-centered search process, where the bees change the position of the food. The major goal of a bee is to find the position of food source with excessive nectar volume. The bee's colony in the ABC algorithm describes three types of bees such as scout bees, employed bees, and onlooker bees[14]. The onlooker bees select their food source based on the employed bee's dance. The employed bees identify their food source and perform a dance on this region to convince their nest helper bees to follow them when they discover a new nectar source. The employed bee becomes a scout bee when their food source is abandoned, to discover a novel food source. The nectar source is chosen by a nest chum whose food source has already recognized.

5. Gravitational Bee Colony Algorithm (GBC)

This paper proposes a hybrid test case selection method called as Gravitational Bee Colony (GBC). This is based on Gravitational Search Algorithm (GSA) and Artificial Bee Colony (ABC) algorithm. The main motive of this method is to lessen the size of test suite and cost, by discovering a set of test cases which gives the same coverage as the original test suite based on some criterion. The proposed technique selects a group of test cases from the already available test suite which can also covers the detected faults.

Bees act as agents to discover a set of test cases. Part of the bees starts foraging with arbitrarily chosen test cases. Then the bee will add fresh test cases on the discovered path. The fault detection capacity is determined by the addition of new test cases. However, after adding a new test case, the bees starts returning to their hive, and transform the obtained information by using GSA[15]. Gravitational Search Algorithm (GSA) is a recently developed search algorithm. The law of gravity and mass interactions is the basics of this algorithm[16]. The heavy masses links to the good solutions for the problems. Solutions are provided by each mass, and by properly adjusting the gravitational and inertia masses, the algorithm is navigated[17]. All the bees are attracted based on gravitational law, and the distance between the two bees can be found by using Motion Law with the help of bee's current velocity and previous velocity. The present velocity of the bee is equivalent to the bee's preceding velocity's sum of coefficient.

Therefore, new set of test cases are generated after the gravitational and motion laws are processed. This procedure should be repeated until any one of the bee find a new group of test cases and additionally covers all faults that are already detected.

The search space is measured as follows

$$x_i = \left(x_i^1, \ldots x_i^d, \ldots, x_i^D\right) \quad (3)$$

Where dimension d of bee i has been shown by x_i^d and the Gravitational forces is calculated as below

$$F_{ij}^d = \frac{G(t)XM_i(t)XM_j(t)}{R_{ij}(t)+\varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \quad (4)$$

Where $G(t)$ is defined as gravitational constant at the $time\ t$. The distance between two bees i and j is R_{ij} and ε is defined as very tiny number.

$$R_{ij}(t) = \left\|X_i(t), X_j(t)\right\|_2 \quad (5)$$

Sum of all forces, forced by all bees is calculated as below

$$F_{ij}^d(t) = \sum_{j=1, j\neq i}^{m} r_j \int_{ij}^{d}(t) \quad (6)$$

Based on Newton's second law, each bee gains accelerating in the direction of $dimension\ d$. The acceleration of $bee\ i$, in the direction of $dimension\ d$, at $time\ t$ is shown by $a_i^d(t)$ and is achieved from following equation

$$a_i^d(t) = \frac{F_{ij}^d}{M_i(t)} \quad (7)$$

The new position of the $bee\ i$, at $dimension\ d$ is also same as the sum of its $current\ position$ and its $velocity\ V$.

$$V_i^d(t+1) = r_i\ X\ V_i^d(t) + a_i^d(t) \quad (8)$$

$$X_i^d(t+1) = r_j\ X\ X_i^d(t)V_i^d + (t+1) \quad (9)$$

Where r_i and r_j are defined as the random numbers with the interval between (0, 1) which have been utilized to retain the random features of the search.

The prerequisite test suite $'T'$ is defined for 'n' number of test cases, and the $subset\ 'S'$ is the result subset. 'S' contains $m\ test\ cases\ (m <= n)$, for which test cases are nominated and considered for increased fault $Coverage$ capacity with time t, where t is the minimum execution time. Finally, the bee's fitness score FS_{abs} is calculated as below

$$FS_{abs} = fiti\begin{cases} \frac{1}{1+f_i} & if\ f_i \geq 0 \\ 1 + abs(f_i) & if\ f_i \leq 0 \end{cases} (10)$$

Where abs defined as absolute value of individual.

Minimizing a test suite is defined as follows

$$J = \sum_{j-1}^{k}\sum_{i=1}^{n}\left\|x_i^{(j)} - c_j\right\|^2 \quad (11)$$

Where $\left\|x_i^{(j)} - c_j\right\|^2$ is the distance between the bees. $x_i^{(j)}$, c_j are indicators of individual $Coverage$ capacity.

Gravitational Bee Colony (GBC)

Step 1: Initialize the population of test cases,

$Test_{case}[Testno][D]$ by the following equation

$Test_{case}[i][j] = r * (ub - lb) + lb$

ub is defined as the Upper bound and lb is defined as the lower bound

Step 2: Initialize

$path_{coverage}[Testno] = 0, trial[Testno] = 0\ \forall\ Test\ cases$

Step 3: $Count = 0$

Step 4: Definethe

$$objective\ value\ (\text{HV}) = \begin{cases} 100 & if\ covers\ any\ new\ path \\ -1 & otherwise \end{cases}$$

Step 5: Define the

$$Fitness\ Function\ (f) = \begin{cases} 1\frac{}{}(\mathbf{HV}+1) & if\ HV > 0 \\ 1 + \mathbf{fabs}(\mathbf{HV}) & otherwise \end{cases}$$

Where HV is defined as hive

Step 6:

$$Set\ path_{coverage}[Testno] = \begin{cases} \mathbf{1} & \mathbf{if\ covers\ any\ new\ path} \\ \mathbf{0} & otherwise \end{cases}$$

Step 7:

$Memorize\ test\ cases\ with\ objective\ value = 100$

Step 8:

$Define\ iteration = 1, count = number\ of\ paths\ covered$

Step 9: Repeat steps 10 to 19 till $maxcycle < MCN$

Employed Bee Phase

Step 10: Create new test cases, $solution[i]$, to the neighborhood $Test_{case}[i][j]$ by the following equation

$Solution[k] = Test_{case}[i][k] + (Test_{case}[i][k] - Test_{case}[n][k]) * (r - 0.5) * 2$

Where n is defined as any test case number in the given test suite, k is the changing parameter of Test *case[i][j]*.

Step 11: Compute fitness and Objective values for the neighbor bee.

Step 12: Apply Gravitational Force process between $Test_{case[i][j]}$ and $solution[i]$.

Step 13: Increase the count as count = count +1 with newly discovered path.

Step 14: If not Increased $trial[i] = trial[i] + 1$

End the Phase of Employed Bee

Step 15: Compute the probability of test cases by the fitness value is as follows:

$$P_i = 1 - \left(\frac{f_i}{\mathbf{f_{max}}}\right)$$

Where P_i values are standardized to $[0, 1]$

Onlooker Bee Phase

Step 16 : If $P_i > r$

Generate new test cases for $solution[i]$ by onlookers bees from the defined test cases $Test_{case[Testno][D]}$, and update the $current\ position$ and $velocity\ V$ of new onlooker bees

Step 17: Else
Modify the test case $Test_{case[i][j]}$

Step 18: Repeat steps 11 to Step 14.

End of Onlooker Bee Phase

Scout Bee Phase

Step19: if $trial[i] > maxtrial$

Step 20: Modify the test case $Test_{case[i][j]}$

Step 21: If the modified test case does not discover any new path, then replace them with the new test case, which is randomly generated by scout waggle dances.

End of Scout Bee Phase

Step 22: $Iter = iter + 1$

After finding the objective functions of test suites, apply the Fuzzy logic prioritization technique for efficient test suite execution. The weights are defined as high, low, medium in prioritization.

$$w_r = \begin{cases} \mathbf{1} & r \in TesterRelevant_r \\ \mathbf{0.5} & TesterPartialRelevant_r \\ \mathbf{0} & TesternonRelevant_r \end{cases} \quad (12)$$

Where $TesterRelevant_r$ and $TesterPartialRelevant_r$, $TesternonRelevant_r$ are requirements r nominated by GBC as partially or relevant or non-relevant. However, the weight w_r is calculated based on the Gravitational Force.

6. Results and Discussion

This section presents the comparison analysis of proposed test suite minimization method GBC with Fuzzy, with other minimization methods as ABC integrated with Fuzzy C-Means (FCM) and PSO (ABCFCM+PSO), Bee Colony Optimization (BCO) and Genetic Algorithm (GA) (BCO+GA)[18] in terms of test suite coverage, time, test suite fault coverage, path coverage and number of iteration.

Table 1. Test suites results

Test Case/ Faults	Test suite 1	Test suite 2	Test suite 3	Test suite 4	Test suite 5	No. of Faults Covered	Execution Time
Test 1	X			X		2	4
Test 2	X	X	X		X	4	7
Test 3	X			X		2	5
Test 4		X	X	X	X	4	11
Test 5		X	X		X	3	6
Test 6	X	X		X		3	9
Test 7	X		X	X	X	4	5
Test 8		X		X	X	3	8

Table 1 shows the proposed work test suite results.

Figure 1 clearly shows that the recommended test case minimization method using GBC with Fuzzy gives the promising statement coverage than ABCFCM + PSO, and BCO + GA optimization methods.

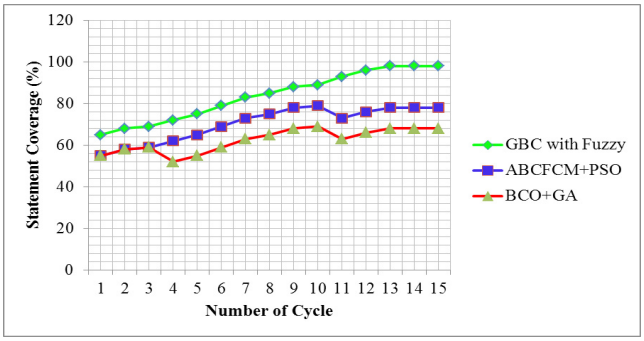

Figure 1. Statement coverage.

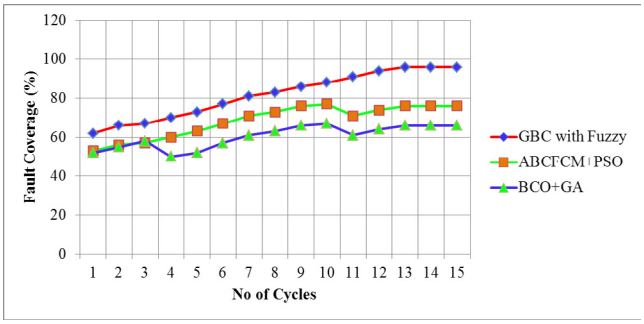

Figure 2. Fault coverage.

It is clearly observed from the Figure 2, that the proposed test suite minimization method utilizing GBC with Fuzzy gives the favorable fault coverage when compared with ABCFCM+PSO and BCO+GA optimization techniques.

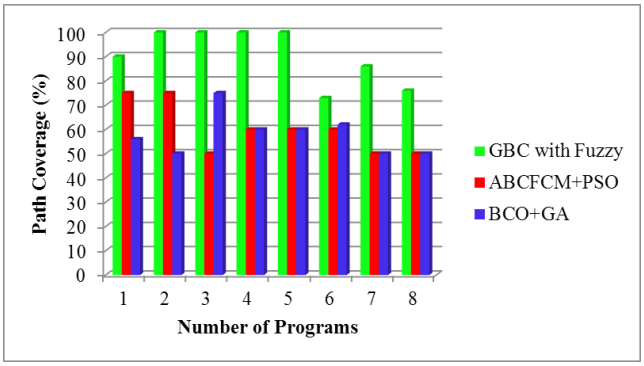

Figure 3. Path coverage.

It is evidently witnessed from Figure 3, that the suggested test case minimization method which uses GBC with Fuzzy gives the increased path coverage than the results of ABCFCM+PSO and BCO+GA test suite optimization methods.

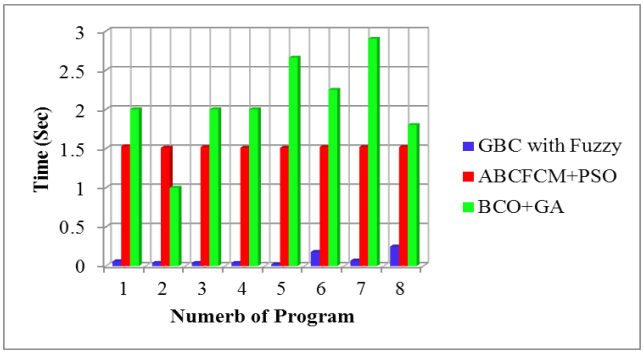

Figure 4. Time.

Figure 4 evidently indicates that the proposed GBC with Fuzzylogic test case reduction technique gives the favorable results in terms of time when compared with ABCFCM + PSO and BCO + GA optimization methods. It took minimum time to process all the test suites.

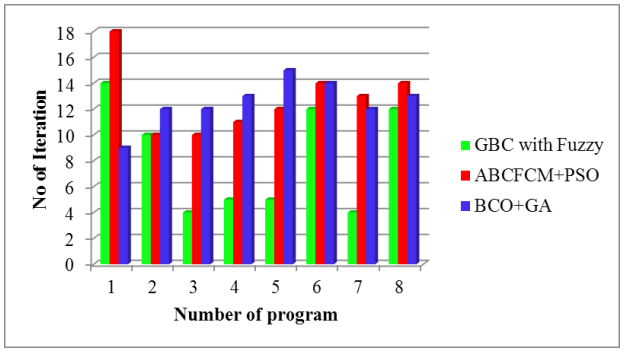

Figure 5. Iteration.

The observation from Figure 5 indicates, that the proposed method using GBC with Fuzzy gives the promising results by giving minimum number of iterations than ABCFCM+PSO and BCO+GA optimization techniques. The proposed work shows the minimum iteration to process all the test suites.

7. Conclusion

This paper mainly focuses on minimizing the test suite by discovering set of test cases that gives the same or better coverage as the original test suite based on some criterion. In this study, the minimization is achieved by using Gravitational Bee Search (GBS) algorithm which is the combination of ABC and GSA. The algorithm searches for optimum solution by calculating fitness values using cov-

erage information. The search process is repetitive until a minimized test suite is identified. The developed technique efficiently identifies and minimizes the test suites. Additionally, the prioritization is done by applying Fuzzy logic for efficient test suite running process. The proposed method gives the promising results in terms of test suite coverage, time, test suite fault coverage, path coverage and number of iterations and it also gives positive feedback in the evaluation process.

8. References

1. Blue D, Segall I, Tzoref-Brill R, Zlotnick A. Interaction-based test-suite minimization. IEEE ICSE; USA. 2013. p. 182–91.
2. Shahabuddin SM, Prasanth Y. Integration testing prior to unit testing: a paradigm shift in object oriented software testing of agile software engineering. Indian Journal of Science and Technology. 2016 May; 9(20). DOI: 10.17485/ijst/2016/v9i20/91223.
3. 3. Parsa S, Khalilian A. On the optimization approach towards test suite minimization. International Journal of Software Engineering and its Applications. 2010; 4(1):15–28.
4. Mudgal AP. A proposed model for minimization of test suite. Journal of Nature Inspired Computing (JNIC). 2013; 1(2):34–7.
5. Prasad S, Jain M, Singh S, Patvardhan C. Regression optimizer a multi coverage criteria test suite minimization technique. IJAIS. 2002; 1(8):5–11.
6. Selvakumar S, Ramaraj N. Multi-objective minimization of test suite and its cost associates using swarm intelligence. International Review on Computers and Software. 2011; 6(2):275.
7. Singh RR. Test suite minimization using evolutionary optimization algorithms: Review. IJERT. 2014; 3(6):2086–91.
8. Joseph AK, Radhamani G. Fuzzy C Means (FCM) clustering based hybrid swarm intelligence algorithm for test case optimization. Research Journal of Applied Sciences, Engineering and Technology. 2014; 87(1):76–82.
9. Agrawal S, Raw RS, Tyagi N, Misra AK. Fuzzy Logic based Greedy Routing (FLGR) in multi-hop vehicular ad hoc networks. Indian Journal of Science and Technology. 2015 Nov; 8(30). DOI: 10.17485/ijst/2015/v8i1/70085.
10. Raman B, Subramani S. An efficient specific update search domain based glowworm swarm optimization for test case prioritization. The International Arab Journal of Information Technology. 2015; 12(6).
11. De Souza LS, Prudêncio RBC, Flávia A, Barros D. A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection. Journal of the Brazilian Computer Society. 2015; 21(19):1–5.
12. Sampath S, Bryce R, Atif M, Memon M. A uniform representation of hybrid criteria for regression testing. IEEE Transactions on Software Engineering. 2013; 39(10):1326–44.
13. Jianga S, Wanga Y, Ji Z. Convergence analysis and performance of an improved gravitational search algorithm. Applied Soft Computing (ASC). Elsevier. 2014; 24:363–84.
14. Mala DJ, Mohan V. ABC tester-artificial bee colony based software test suite optimization approach. IJSE. 2009; 2(2):1–33.
15. Malhotra R, Anand C, Jain N, Mittal A. Comparison of search based techniques for automated test data generation. International Journal of Computer Applications. 2014; 95(23):1–5.
16. Khajehzadeh M, Eslami M. Gravitational search algorithm for optimization of retaining structures. Indian Journal of Science and Technology. 2012 Jan; 5(1). DOI: 10.17485/ijst/2012/v5i1/30937.
17. Tavakkolai H, Yadollahi N, Yadollahi M, Hosseinabadi AAR, Kardgar M. Sensor selection wireless multimedia sensor network using gravitational search algorithm. Indian Journal of Science and Technology. 2015 Jul; 8(14). DOI: 10.17485/ijst/2015/v8i14/68808.
18. Suri B, Mangal I, Srivastava V. Regression test suite reduction using an hybrid technique based on BCO and genetic algorithm. IJCSI. 2009; 2(1-2):165–72.