Test Case Prioritization for Regression Testing Using Immune Operator

Angelin Gladston[1], Khanna Nehemiah[1], Palanisamy Narayanasamy[2], and Arputharaj Kannan[2]
[1]Ramanujan Computing Centre, Anna University, India
[2]Department of Information Science and Technology, Anna University, India

Abstract: *Regression testing is a time consuming, costly process of re-running existing test cases. As software evolves, the regression test suite grows in size. Test case prioritization techniques help by ordering test cases such that at least the test cases which cover the changes made in the software are executed amidst resource and time constraints. Genetic Algorithm (GA) has been widely used for test case prioritization problem, however it has low convergence problem. In this work, the Immune Genetic Algorithm (IGA) is applied for test case prioritization, so that test case prioritization converges earlier. Our contributions in Immune Prioritization Algorithm (IPA) include a method for vaccine selection, zero drop function and probability selection function. The prioritized result of IPA is evaluated against GA and the statement coverage, decision coverage and block coverage of the test cases prioritized using IPA are found to have improved. Further, IPA showed improved average fitness value as well as optimal fitness value compared to genetic algorithm.*

Keywords: *Immune operator, vaccine, test case prioritization, regression testing, GA, IPA.*

Received July 3, 2012; accepted April 29, 2013; Published online December 23, 2015

1. Introduction

Changes to software are inevitable. Evolving software needs to be tested after the incorporation of changes. Hence, regression testing is carried out by re-running all test cases in order to test the software. With limited resources and time, the re-execution of all test cases is not possible. Hence, test case prioritization is used to make regression testing more meaningful. This is accomplished since test case prioritization rearranges the test cases so that, the test cases which cover the newly incorporated changes can be brought first for further execution.

In search based software engineering, the software engineering problems are modeled as optimization problems. Hence, for test case prioritization problem, metaheuristic search techniques which find optimal or near optimal solutions for optimization problems can be applied [4]. The application of metaheuristic search techniques namely, Hill climbing [13], Tabu Search [11] and Genetic Algorithm (GA) [13] to test case prioritization were already carried out. In this work, the application of Immune Genetic Algorithm (IGA) for test case prioritization problem is examined. An Immune Prioritization Algorithm (IPA) which employs a vaccine selection function to select vaccines for every chromosome is proposed. It utilizes a zero drop function to retain the chromosomes that are fit enough and prioritizes them using the probability selection function.

The purpose of using immune concepts in GA is to make use of the local characteristic information to arrive at an optimal solution. In test case prioritization problem, the test cases of maximum coverage are expected to come first. To attain that every chromosome is expected to accommodate test cases which will maximize its fitness value. This is accomplished by having a test case which comes earlier in order rather than a test case which comes late in order of same coverage. This local characteristic information is used, to avoid repetitive work and to overcome the blindness in action. Immune concept helps restrain from the degeneration arising from evolutionary process. This results in steady increase of fitness value. The immune concept [7] was realized using two steps: Vaccination and immune selection. Vaccination is a process of modifying genes on some bits according to some prior knowledge so as to gain higher fitness with improved probability. The correctness of the information used in selecting a vaccine plays a vital role in the performance of the algorithm. Immune selection ensures whether the offspring is having improved fitness value compared to parent or not. If the offspring is having low value then the parent is retained for next generation.

The organization of the paper is as follows: Section 2 describes works related to the test case prioritization problem and works related to the application of IGA. Section 3 describes the proposed IPA. Section 4 presents the implementation details and experiments conducted. Section 5 reports the results and discussion and section 6 concludes the work.

2. Related Works

Application of heuristic algorithms namely, Greedy algorithm, Additional Greedy algorithm, 2-Optimal algorithm, Hill climbing and GA were assessed by Zheng *et al*. [13]. Five 'C' programs were used. Average Percentage Block Coverage (APBC), Average Percentage Decision Coverage (APDC) and Average Percentage Statement Coverage (APSC) were the coverage metrics used. The data analysis indicated that GA and additional greedy algorithm outperformed greedy algorithm. Results show that APBC for genetic is 93%, Additional Greedy is 92% and that of greedy is 90%. A pareto approach to prioritize test suites based on multiple objectives, such as code coverage, execution cost and fault detection history was described by Yoo and Harman [12]. The objective was to find an array of decision variables (test case ordering) that maximize an array of objective functions. The pare to efficient GA produces subsets of test cases that can be executed in fewer than 200 units of cost, something for which the additional greedy algorithm was not capable.

Hyunsook *et al*. [5] has evaluated the effects of time constraints on the costs and benefits of prioritization techniques. Metrics like Average Percentage Faults Detected (APFD) was not sufficient to assess time-constrained techniques. Further, rather than considering the costs of applying techniques and of utilizing them on single system releases, the costs across entire system lifetime were considered by the metric. Their proposed evolution-aware economic model for regression testing captured the above factors. This model helped to select testing processes and prioritization techniques that are most appropriate for their organizational and process contexts. Five Java programs were used and kruskal-wallis tests showed that faultiness levels, FL1, FL2, and FL3 have significant effects in all cases. Faultiness level, FL1 involves cases in which mutant groups contain 1 to 5 faults. Faultiness level, FL2 involves cases in which mutant groups contain 6 to 10 faults. Faultiness level, FL3 involves cases in which mutant groups contain 11 to 15 faults. As faultiness levels move from FL1 to FL2 benefit of the prioritization technique improve in 12 out of 20 cases. As faultiness levels move from FL2 to FL3 benefit of the prioritization technique improve in all 20 cases.

Jiao and Wang [7] introduced IGA based on the theory of immunity in biology which constructs an immune operator accomplished by vaccination and immune selection. IGA restrains from the degeneration phenomena effectively during the evolutionary process. It was shown that IGA improves the searching ability and adaptability, greatly increasing the converging speed using travelling salesman problem. As a comparison between IGA and GA, 20 initial populations were generated and vaccines were produced. Distribution of individuals without vaccine, with vaccine, variation of optimal fitness with the reproduction of the offspring and variation of average fitness with the reproduction of the offspring were observed. They conclude that IGA finds the global optimum ($f(x)_{max}$=19.8949; $x_{optimal}$=0.1275) after 12 iterations, while GA finds the local optimum ($f(x)$= 19.8903; x=0.1273) after 53 iterations, where f is the fitness function on x and x is the search space.

Bouchachia [2] incorporated IGA as an advanced method for solving the test data generation problem. Immune filtering operator is used after the genetic mutation operator. The application of IGA in the context of software test data generation was investigated using some benchmark programs. Three input programs were used. On an average, testing coverage for IGA is 98.95% but for GA it is 94.58%. Test case generation problem and use of metrics for extensive generation is studied by Izzat and Mohammed [6]. Liu and Peng [9] proposed an Immune Particle Swarm Optimization (Immune-PSO) Algorithm which solves prematurity problem in PSO because of the decrease of swarm diversity. PSO is a widely used [3] meta heuristic algorithm. PSO algorithm is combined with Immune clone selection algorithm. The typical benchmark functions are performed. The numerical simulation results showed that the improved algorithm maintains swarm's diversity, speeds up convergence speed, and also escapes from local extreme. Results show that Immune-PSO algorithm outperforms GA. The mean squared error of GA is 0.009 while that of Immune-PSO is almost 0. Further the running time of GA is six times as much as the Immune-PSO.

IGA has been used for random test pattern generation by Azimipour *et al.* [1] for testing Very Large Scale Integrated (VLSI) circuits. The problems with GA are that they may drop into local optimal solutions, or they may find the optimal solution by low convergence speed. To overcome these problems IGA is used. Calculating test vector density, activating test vectors and suppressing test vectors based on density are the various immune operations used. The application of few main characteristics of IGA to GA for test pattern generation improved the test size with a factor of about 25% in comparison with Non-Immune algorithms.

Comparing to the works described in the related work section, the work described in this paper is different in the following ways: First, for vaccination, in the related work discussed [7] for all the candidates the same selected gene was used as vaccine. The selection strategy used in [7] involves two steps: Analyzing the pending problem and collecting the characteristic information and producing vaccines according to the characteristic information. However, in this paper, for every chromosome, appropriate vaccine was selected and used. Since, the purpose of vaccine is to improve the fitness value of the chromosome, the test case which comes late in the order, thereby resulting in low fitness value is replaced with the vaccine, which is selected to be the test case of same coverage criteria, but which comes earlier in the order. Thus, each chromosome is vaccinated with a vaccine suitable for that. Further, in the work discussed in [10] genes are randomly selected from the gene pool developed and finally inoculated into genes. But in this paper, rather than randomly selecting gene from gene pool, the best gene is selected and vaccinated. The selection and vaccination are discussed in detail in section 3.

Second, immunization is tailored to incorporate two additional functions, a zero drop function and another for probability selection along with the existing functions namely, immune test and probability calculation function used in the literature. The zero drop function checks for any zero as the fitness value gets calculated and if any zero exists, then drops the corresponding candidate. The probability selection function rearranges the chromosomes using the calculated probability to prioritize test cases better, thus helps in regression testing. The immunization process is discussed in detail in section 3.

3. Experimental Design

IGA has already been used for travelling sales man problem [7], test data generation [2] as well as for test pattern generation [1]. GA has been extensively used to solve test case prioritization problem [12, 13]. However, it has delayed convergence problem. Hence, an IPA, which uses IGA, is used for test case prioritization problem. IPA utilizes the two steps of IGA: vaccination and immunization. The chromosomes are vaccinated first and then the vaccinated chromosomes undergo immunization that is immune test. In vaccination, the problem is analyzed and a vaccine is selected using the characteristic information collected. Test case of same coverage but which comes earlier in the order was chosen as vaccine and vaccination is performed using that. The immunization process is incorporated with a zero drop function and a probability selection function along with existing immune test and probability calculation function. The zero drop function checks for any zero, as the fitness gets calculated and drops the corresponding candidate. The probability selection function rearranges the candidates, using the calculated probability.

3.1. Immune Prioritization Algorithm

The purpose of vaccination is to gain higher fitness. The vaccine selection function is tailored for test case prioritization problem so that the fitness value of chromosome can be improved. First the test case prioritization problem is analyzed and the characteristic information is obtained. Vaccines are produced based on the characteristic information. The theme of test case prioritization problem is to prioritize the chromosomes based on coverage. The chromosome with best coverage is to be given first priority, then the chromosome with next coverage and so on. Here, condition coverage metric is used and fitness for each chromosome is computed using that. The ultimate aim is to ensure that all requirements are tested with minimum number of test cases chosen from the top of the prioritized test suite. Hence, the problem is to find the chromosomes with maximum fitness value, which satisfies maximum requirements. Consider a chromosome has six genes, $c=\{t_1, t_2, t_3, t_4, t_5, t_6\}$. Each gene is a test case. Consider there are five requirements, r_1, r_2, r_3, r_4 and r_5. Suppose, t_1, t_2 satisfy r_1, r_2 respectively, t_3 and t_4 satisfy r_3 whereas t_5 and t_6 satisfy r_5. Analyzing the chromosome shows light that we have more test cases satisfying same requirements but not all requirements are satisfied by the chromosome. Therefore, if we can have a test case satisfying the requirement, r_4, that will help to overcome the problem. Based on these characteristic information obtained, vaccines are produced. The vaccine is chosen such that the test case which comes late in the order is identified and replaced with a test case covering same criteria but which comes earlier in the order. Thereby, the fitness value of the chromosome is raised further and it is made to satisfy more number of requirements. Consider an initial population, $T\_init=\{c_1, c_2, \dots, c_i, \dots, c_n\}$ where c_i is a chromosome of test cases $\{t_a, t_b, t_c, \dots\}$, for every i from 1 to n, then the vaccine function, $V(c_i)$ is applied on all chromosomes. Chromosome comprises of genes which are test cases here. Hence, in the process of vaccination, that is modifying genes, a candidate t_i is identified and vaccinated with another t_j resulting in higher fitness value as shown:

$$V(c_i)=\{t_i \leftarrow t_j\}$$

where t_j comes earlier in order than t_i and both t_j and t_i satisfy same coverage requirements.

Thus, the selected vaccine helps to improve the fitness value of chromosomes. The vaccine is selected for each chromosome and an appropriate vaccine is selected for each chromosome analyzing the ways to improve its fitness value. Applying the selected vaccine makes the chromosome more fit. Further, in this work the test case which comes earlier in order with same coverage is chosen as vaccine rather than developing a pool and randomly selecting from it [10]. This is followed by immunization. The two usual immune selection operations employed are an immune test and probability calculation. First, immune test is applied. Immune test ensures whether the offspring is having improved fitness value compared to parent or not. If the offspring is having low value then the parent is retained for next generation. Thus, degeneration is avoided. Otherwise, the offspring is chosen for next generation. In addition to the usual immune selection functions, two new functions, a zero drop function and a probability selection function are applied. The zero drop function works on the candidates, applied with immune test. As the fitness value gets computed it searches for any zero. Occurrence of zero indicates that the particular candidate lacks coverage. Hence, such candidates are dropped, to yield a test suite with improved coverage. Then, probability selection function is applied based on the calculated probability. The selection function renders the test cases prioritized based on condition coverage as the operator keeps selecting test cases based on the probability computed out of condition coverage.

The steps carried out in IPA are as follows:

- *Step* 1: Create initial random population, *T_init*.
- *Step* 2: Select vaccines using prior knowledge vaccines for each chromosome are selected so as to improve its fitness value. Possibilities of increasing the fitness value of chromosomes are analyzed and a vaccine is selected such that the fitness value of c_i' is greater than that of c_i. (our contribution).
- *Step* 3: If the current population contains the optimal individual, then the iteration stops; or else continues from step 4.
- *Step* 4: Perform crossover on the k^{th} parent and obtain the resulting T_cross_k.
- *Step* 5: Perform mutation on T_cross_k and obtain the resulting T_mutate_k.
- *Step* 6: Perform vaccination on T_mutate_k and obtain the resulting T_vacc_k, vaccination is applied on all chromosomes of T_mutate_k to obtain T_vacc_k. Test case with less fitness value is identified and vaccine is applied, thereby it gets replaced by test case of higher fitness value. $V(c_i)=\{t_i \leftarrow t_j\}$ such that: $t_i \in c_i$ and t_j comes earlier in order than t_i as well as both t_j and t_i satisfy same coverage criteria.
- *Step* 7: Perform immune selection on T_vacc_k and obtain the next parent and then go to step 3.
- *Step* 8: Perform zero drop function, fitness is computed for all chromosomes using the fitness function used in [8], $F(c_i)=1-((tc_1+tc_2+, \dots, tc_j+, \dots,+tc_n)/nm)+1/2n$, where n is the number of test case sequences, m is the number of conditions in the input program and tc_j is the location of the first test case in the sequence which covers the j^{th} condition.

  Zero drop function checks for any zero, whenever a tc_i is zero, it indicates that the corresponding coverage criteria is not met, hence that chromosome is dropped (our contribution).
- *Step* 9: Perform probability selection function to render the optimal individual prioritized, probability is calculated using the mathematical model in [1] for the remaining chromosomes as given by, probability, $P(c_i)=F(c_i)/\sum F(c_i)$ where $i=1, \dots, n$.

  Probability selection function uses Prioritize function to reorder c_i*'s* based on $P(c_i)$ (our contribution).

Algorithm 1 can be stopped either at the maximal iterative number or when the fitness begins to remain steady for a preset number of iterations.

Algorithm 1: Immune prioritization.

```
$T\_init=\{c_1, c_2,\dots, c_n\}$
$c_i$ is a chromosome of test cases $\{t_a, t_b, t_c,\dots\}$
#Vaccine Selection
for($i$=1 to $n$)
Compute $V(c_i)$ such that $F(c_i')>F(c_i)$
# $c_i$ is chromosome before vaccination
# $c_i'$ is chromosome after vaccination
#$F(c_i)$ and $F(c_i')$ gives fitness value before and after vaccination
while(not optimal)
{
Crossover ($c_k$, $c_{k+1}$)
#$F(c_k)$ and $F(c_{k+1})$ are the highest among other candidates
Mutate ($c_k$,$c_{k+1}$)
#Vaccination
for($i$=1 to $n$)
Apply $V(c_i)$ on all chromosomes
#Immune Selection
If $F(c\_vacc_{k+1})>F(c\_mutate_{k+1})$ and $F(c\_vacc_k)>F(c\_mutate_k)$
   then
    select offspring to be next parents
else
     retain parents for next generation
}
#Zero drop function
for($i$=1 to $n$)
            If any $tc_i$=0 in $F(c_i)$ then
                        drop $c_i$ from $T\_vacc_k$
 #Probability selection function
for($i$=1 to $n$)
          $P(c_i)=F(c_i)/\sum F(c_i)$
# $c_i$ with maximum probability is ranked first.
Prioritize ($T\_vacc$)
returns prioritized test suite
```

4. Experiments

A PL/SQL procedure named *First_Run* utilized for the study is part of the Pay Roll Management System used in Anna University. The system which was developed using Oracle 10g comprises of thirty three relations. The First_Run procedure populates the earnings_trial_run relation and deductions_trial_run relation with new values. The earnings_trial_run relation is meant for storing the earnings of employees applicable in a month and deductions_trial_run relation is meant for storing the deductions of employees applicable in a month. These two relations are used to make report generation easy. The earnings_trial_run relation is populated with earnings from earnings relation. Deductions_trial_run relation is populated with various deductions that are drawn from the sixteen relations, employee_gpf which stores employee provident fund details, employee_cps which stores employee contributory pension scheme details, deduction_specific which covers deduction details of additional house rent, electricity charge, cooperative society, vehicle maintenance, as well as employees recreation club. In addition to these, deductions from variable_ded_std which captures details of family benefit fund, mediclaim, government health insurance scheme, as well as professional tax, cps_recovery which stores employee contributory pension scheme recovery details, spf which stores special provident fund details, court_recovery, income_tax_deduction, gpf_loan, house_loan, loan_sanction, rop which stores recovery of over payment details, bank_loan, licpolicy, pli which stores postal life insurance policy details and rec_deposit which stores details of recurring deposits.

Test cases were generated randomly for the input program, First_Run. The generated test cases were grouped based on the coverage information of the test cases. Table 1 provides the details of the program used for the study. Test cases were grouped into chromosomes to form the initial population *T_init*. IPA is tested with different number of iterations 20, 25, 29, 30, 31, 35, 40, 45, 50, 100, 120, 200, 400, 600, 800, 1000 as well as 1200 and the best average fitness value is achieved at 100 iterations.

Table 1. Program used for the study.

Program	Lines of Code	Test Suite Size
First Run	693	520

The test cases generated were prioritized using GA and IPA. GA used by Zheng *et al.* [13] for test case prioritization was used for evaluating IPA.

5. Results and Discussion

To evaluate the performance of the proposed approach, the results obtained by the IPA were compared with that of GA. The results were evaluated using three metrics, APSC, APDC and APBC. Further, different percentages of the test suite, namely 5%, 10%, 15%, 20% and 25% were considered. That is, 5% means first 5% of test cases were chosen for execution and 10% means first 10% of test cases were chosen for execution and so on. For evaluation, the algorithms were executed 5 times. In the executions, different orderings of prioritized test cases were obtained and the results were normalized by calculating the average.

The application for generating test cases and collecting coverage information along with the test case prioritization algorithms used in the experiments were implemented in Java. All applications were run on eclipse environment.

The results are presented in Tables 2, 3 and 4. Table 2 presents the APSC values for IPA and GA with 5%, 10%, 15%, 20% and 25% of Test suite. Table 3 presents the APDC values for IPA and GA with 5%, 10%, 15%, 20% and 25% of Test suite and Table 4 presents the APBC values for IPA and GA with 5%, 10%, 15%, 20% and 25% of Test suite. Different percentages of test suite from 5% up to 25% were chosen for analysis since both the algorithms showed varying coverage until they achieve 100% statement, decision as well as block coverage with 25% of test suite.

Table 2. APSC with various percentages of test suite.

Test Suite	GA	IPA
5%	87.15	86.86
10%	91.63	92.06
15%	95.09	97.25
20%	97.71	100
25%	99.71	100

Table 3. APDC with various percentages of test suite.

Test Suite	GA	IPA
5%	95.65	99.81
10%	98.12	100
15%	100	100
20%	100	100
25%	100	100

Table 4. APBC with various percentages of test suite.

Test Suite	GA	IPA
5%	73.79	78.62
10%	80.00	90.34
15%	85.51	99.31
20%	96.55	100
25%	100	100

Analyzing the results obtained from the experiments, which are detailed in Tables 2, 3 and 4, the following conclusions have been arrived. First, for all the percentages of test suite analyzed IPA significantly outperforms GA. This shows the applicability of Immune operator for prioritization problem. Further, considering APSC, APDC and APBC in general, IPA performs better than GA except 5% for APSC.

Second, as the percentage of test suite size increases from 5%, IPA behaves better than GA with APSC, APBC as well as APDC. There is a significant increase in the APSC of IPA from 5% to 20%. When APDC is considered, IPA outperforms GA with APDC of 99.81% even with 5%. Similarly for APBC, IPA achieves more than 90% coverage even with 10%. This is because the candidates having low fitness function have been vaccinated to get their fitness value improved. Because of that IPA performs better even with 10% test suite attaining more than 90% coverage in all three. Table 3 shows that, when 15% test cases were chosen for execution, then 100% performance is achieved with IPA. What we infer here is, when there are resource constraints and not all test cases can be executed, then it is where these algorithms play a vital role and it is where IPA can be opted for. That is, when only few percentages of test cases can be executed then IPA can be utilized better.

Figures 1, 2 and 3, demonstrates the performance comparison of the First_Run among the two algorithms, IPA and GA for the metrics APSC, APDC and APBC. The different percentages of test suite used, namely 5%, 10%, 15%, 20% and 25% are marked along x axis. The figures clearly show that IPA performs well compared to GA. Hence, IPA is a better solution for prioritization problem. Further, the main issue with regression testing is resource constraint. This IPA addresses the resource constraint issue nicely. Figures 1, 2 and 3, show clearly that the immune prioritization provides 97.25% statement coverage whereas Genetic provides 95.09% statement coverage and 99.31% block coverage whereas genetic provides

85.51% when only 15% test cases can be executed because of resource constraints. In APSC there is 2.29% increase and in APBC there is 13.8% increase, which shows clearly the performance of IPA. IPA achieves 100% APSC and APBC even with 20% of test suite whereas GA needs 25% or more. IPA achieves 4.16% increase in APDC when compared with GA and achieves 100% coverage with just 10% of test cases.

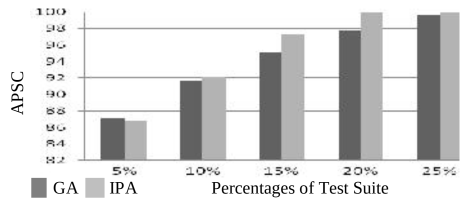

Figure 1. APSC with various percentages of test suite.

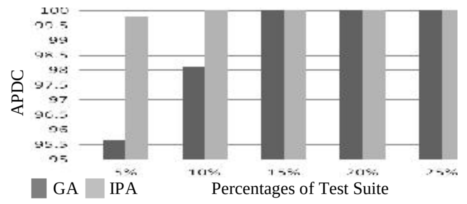

Figure 2. APDC with various percentages of test suite.

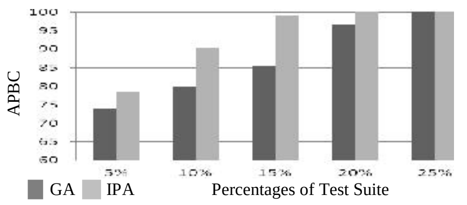

Figure 3. APBC with various percentages of test suite.

GA and IPA are executed for different iterations and the average fitness values and optimal fitness values are shown in Figure 4. Each generation of IPA is a little longer because of the vaccination and immunization operations compared to that of GA, but still IPA outperforms GA since IPA arrives at average fitness value of 80.3 in just 4 generations whereas GA reaches an average fitness value of 79.2 in 45 generations. Moreover, IPA obtains optimal fitness value of 99 and remains steady. Further IPA attains an average fitness value of atleast 86.2 in all iterations whereas GA attains a maximum of 79.2. Thus IPA helps in achieving better prioritization and the algorithm converges quickly because of the zero drop and probability selection functions utilized.

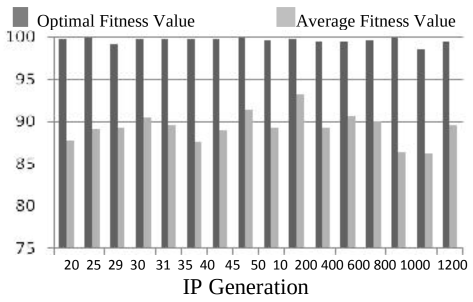

Figure 4. Optimal and average fitness value with various IPA iterations.

6. Conclusions

All these experiments have shown improved APSC, APDC and APBC for IPA compared to GA. The experiments conducted with different number of generations clearly shows IPA is capable of achieving optimal fitness value of 99 and remains steady. IPA attains the average fitness value of atleast 86.2 whereas for GA it is a maximum of 79.2. This clearly shows that IPA needs less number of iterations to reach steady fitness value compared to GA, which paves way for arriving at a better prioritized test suite. APSC of IPA is 2.29% more compared to GA for 15% of test cases. APDC of IPA is 4.16% more compared to GA for 10% of test cases and APBC of IPA is 13.8% more compared to GA for 15% of test cases. Thus, IPA outperforms GA. Further, the difference between APSC, APDC and APBC of Immune Prioritization with that of genetic is excellent for all percentages of test cases. IPA performs well with incremental changes wherein we render having an enormous test suite. Further, it helps ensuring that the test cases with maximum fitness value and coverage are considered for testing first. This shows that the applicability of IPA will be more when more unfit test cases are involved. Hence, for applications where many test cases evolve along with software and become unfit as far as coverage is concerned, this IPA can be best utilized for prioritizing test cases.

References

[1] Azimipour M., Bonyadi M., and Eshghi M., "Using Immune Genetic Algorithm in ATPG," *Australian Journal of Basic and Applied Sciences*, vol. 2, no. 4, pp. 920-928, 2008.

[2] Bouchachia A., "An Immune Genetic Algorithm for Software Test Data Generation," *in Proceedings of the 7th International Conference on Hybrid Intelligent Systems*, Kaiserlautern, pp. 84-89, 2007.

[3] Fahd M., Mohiy H., Kamel M., and Khalid A., "A New Image Segmentation Method Based on Particle Swarm Optimization," *The International Arab Journal of Information Technology*, vol. 9, no. 5, pp. 487-494, 2012.

[4] Glover F. and Kochenberger G. *Handbook of Meta Heuristics*, Springer, Berlin, Germany, 2003.

[5] Do H., Mirarab S., and Rothermel G., "The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments," *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 593-617, 2010.

[6] Alsmadi I. and Al-KabiIzzat M., "GUI Structural Metrics," *the International Arab Journal of Information Technology*, vol. 8, no. 2, pp. 124-129, 2011.

[7] Jiao L. and Wang L., "A Novel Genetic Algorithm Based on Immunity," *IEEE*

Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 30, no. 5, pp. 552-561, 2000.

[8] Jun W., Yan Z., and Chen J., "Test Case Prioritization Technique based on Genetic Algorithm," *in Proceedings of International Conference on Internet Computing and Information Services*, Hong Kong, pp. 173-175, 2011.

[9] Liu F. and Peng B., "Immune-Particle Swarm Optimization Beats Genetic Algorithms," *in Proceedings of the 2nd WRI Global Congress on Intelligent Systems*, Wuhan, pp. 233-236, 2010.

[10] Lu J. and Xie M., "Immune-Genetic Algorithm for Traveling Salesman Problem," available at: http://cdn.intechopen.com/pdfs-wm/12405.pdf, last visited 2010.

[11] Srivastava P., Vijay A., Barukha B., and Sengar P., Sharma R., "An Optimized Technique for Test Case Generation and Prioritization Using Tabu Search and Data Clustering," *in Proceedings of the 4th Indian International Conference on Artificial Intelligence*, pp. 30-46, 2009.

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

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

Angelin Gladston is a Research Scholar in Ramanujan Computing Centre, Anna University, India. She is working as an Assistant Professor in Department of Computer Science and Engineering, Anna University, Chennai. Her research interests include software engineering, software testing and data mining.

Khanna Nehemiah is working as an Associate Professor in Ramanujan Computing Centre, Anna University, India. His research interests include software engineering, database management systems, data mining and medical image processing.

Palanisamy Narayanasamy is working as a Professor in Department of Information Science and Technology, Anna University, India. His research interests include networks, mobile computing and software engineering.

Arputharaj Kannan is working as a Professor in Department of Information Science and Technology, Anna University, India. His research interests include software engineering, database management systems, data mining and artificial intelligence.