Asian Journal of Information Technology 15 (8): 1405-1410, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

GNA Based Test Case Prioritization

[1]R. Uma Maheswari and [2]D. Jeya Mala
[1]Department of Computer Applications, K.L.N.College of Engineering, Sivagangai, India
[2]Department of Computer Applications, Thiagarajar College of Engineering, Madurai, India

Abstract: Changes in software can affect some of its behavior that had been implemented until that point. To discover such a problem, the perfect solution is testing the entire system completely again but there is a lack of time or resources for doing that. A different approach is to order the test cases and execute a subset of them which guarantees that the essential tests are executed first. Such an approach is known as regression test case prioritization (TCP). In this study, a new test suite prioritization technique which is a variant of Genetic Algorithm is presented. The proposed approach is based on Global Neighborhood Algorithm (GNA) which combines local search and global search. In the proposed system, the chromosomes represent the test case Sequence and gene represents the test case numbers. The aim is to find a test case sequence with 100% Fault Coverage in minimum time. Examination on whether the proposed approach overtakes existing test prioritization approach based on Genetic Algorithms (GA) for software test Prioritization is carried out. From the analysis of the results of the experiments, it is inferred that: computation time of GNA based Approach is very negligible over GA based Approach and GNA based approach is best in finding global optimal solution.

Key words: Regression testing, test case prioritization, global neighborhood algorithm, genetic algorithm, approach

INTRODUCTION

The affirmation with respect to any alteration in the product is given by the procedure called regression testing. It ensures that modifications have not influenced functional characteristics of software. It is very costly method to be utilized. Some techniques such as test selection reduction and test prioritization (Yu and Lau, 2012) have been proposed by researchers for viable cost minimization in regression testing. Test Case prioritization in regression testing orders the test cases such that highest priority test case is to be executed first and so on, according to selected criteria such as Statement, Path, Branch and Fault coverage.

In TCP, we have to find the ideal sequence's that will give the best fitness. In the event that every possible combinations are to be checked, then the aggregate number of the combinations will be N!. In case of Large Test suites, TCP is observed to be NP-hard with no deterministic solution and has exponential complexity in the worst case. It will be hard to tackle the TCP issue by utilizing conventional mathematics or using numerical induction techniques. So, Meta heuristic techniques pick up prominence in TCP Problem Solving. Meta-heuristic search techniques are high-level frameworks which utilize heuristics in order to discover solutions for combinatorial optimization at sensible computational expense.

A new optimization technique GNA is a population based and derivative free algorithm like other evolutionary optimization. It is developed to optimize combinatorial problems. It will work to find the global optimal value among the local optima by switching between exploration and exploitation. It has been tested and proved to be effective to solve traveling salesman problem (Alazzam and Lewis, 2013). In our approach, the functionality of the GNA is extended to solve TCP to reduce the manual work and improves the confidence on the software by testing it with the coverage of the given software. In our approach Fault coverage is used as test adequacy criteria. Since an improved rate of fault detection can provide earlier feedback on the system under test, enable earlier debugging and increase the likelihood that if testing is prematurely halted, those test cases that offer the greatest fault detection ability in the available testing time will have been executed.

Literature review: Many prioritization techniques have been described in the research literature (Krishnamoorthi and Mary, 2009; Panigrahi and Mall, 2013; Jacob and Ravi, 2013; Maheswari and Mala, 2013)

Corresponding Author: R. Uma Maheswari, Department of Computer Applications, K.L.N.College of Engineering, Sivagangai, India

for TCP. Several meta-heuristic search techniques such as Genetic Algorithm (GA) (Wang *et al.*, 2015) Ant Colony Optimization (ACO) (Mala and Mohan, 2009) and Cuckoo Search (Nagar *et al.*, 2015), etc. Hybrid Techniques (Maheswari and Mala, 2015; Suri *et al.*, 2011) have also been proposed for Prioritization.

The important issues noticed in literature study of Meta heuristic based TCP are:

- The results of Genetic algorithm are not stable and found to stick at local optima often. The convergence is slow and non-explicit memorization of best individuals
- The drawbacks of Ant Colony Optimization are long length test sequences and repetition of nodes within the same sequence without any improvement in test adequacy criteria
- In Tabu search based approach, more amount of memory is required

The earlier said issues of existing work have made us to concentrate on an alternate approach which has the merits of population based approaches without the issue of local optima. The new Optimization techniques GNA has provided us a lot of hope as it is used to solve Traveling Salesman Problem successfully which resembles TCP Problem Solving. In the light of the above consideration, we applied GNA for software test suite Prioritization.

Gna algorithm-an introduction: GNA is a population based and derivative free algorithm like other evolutionary optimization. It is developed to solve combinatorial optimization problems. The combinatorial optimization problems may contain more than one local and global optimal value within the huge search space. By Transiting between exploration and exploitation, GNA will work to find best optimal value. Exploration searches the whole huge solution space. Exploitation focuses in the neighborhood of the best solution.

In GNA, set of (m) solutions are first arbitrarily created from the huge search space. Then the fitness for the above solution will be found using the objective function .The solutions are then ordered by their fitness obtained. Let S1 is the best solution among them. It is taken as a good measure for the local optimal solution and also as the current best solution.

Next population is formed by including m/2 solution from neighborhood of best solution S1 and m/2 solution from the entire search Space. Solution from huge search space allows exploration of search space. Solutions from neighborhood allow exploitation. It is done because the function that needs to be optimized could have more than one local optima and it may stuck at one of these local optima.

The best solution in the newly generated population is calculated. It is then compared with the current best solution S1 and replaces it if found better. The process is then repeated until a specified number of iterations (t), or when there is no further improvement on the final value of the optimal solution.

Pseudocode:

```
Define fitness function (f)
Initialize GNA parameters: Population size m, No of Iteraions t
Create (m) random solutions from the entire search space
Calculate the fitness of m solutions
Optimal solution= the best solution.
i=1
While i<t
Generate m/2 solutions from the neighborhood of the best solution
Generate m/2 solutions from the search space
Find the best solution from the (m) generated solution
If best solution is better than optimal solution
Optimal solution=best solution
End If
i++
End DO
End DO
```

Proposed approach- GNA based TCP: The GNA algorithm was used to solve the Test Case Prioritization Problem. The Framework for the GNA based TCP is given in Fig. 1. The TCP Problem consists of finding a sequence of test cases in a test suite .In order to optimize the TCP Problem, the optimal sequence of test cases that maximizes the fault coverage in minimum time has to be found.

Problem statement: Given a test suite, T, the set of permutations of T , PT and f , a function from PT to the set of real numbers, Problem: find T ‘ ª PT such that:

$$(\forall T'') (T'' \in PT) (T' \neq T'') [f(T') \geq f(T'')]$$

In the above definition, PT represents the set of all possible prioritizations (orderings) of T, and f is a function that applies to any such ordering, yields an award value for that ordering.

Genetic coding: To apply GA for any optimization problem, one has to think a way for encoding solutions as feasible chromosomes so that the crossovers of feasible chromosomes result in feasible chromosomes.

Since test cases are to be ordered in some sequence, Permutation encoding is selected. The Chromosome represents the test case Sequences. Each gene in a

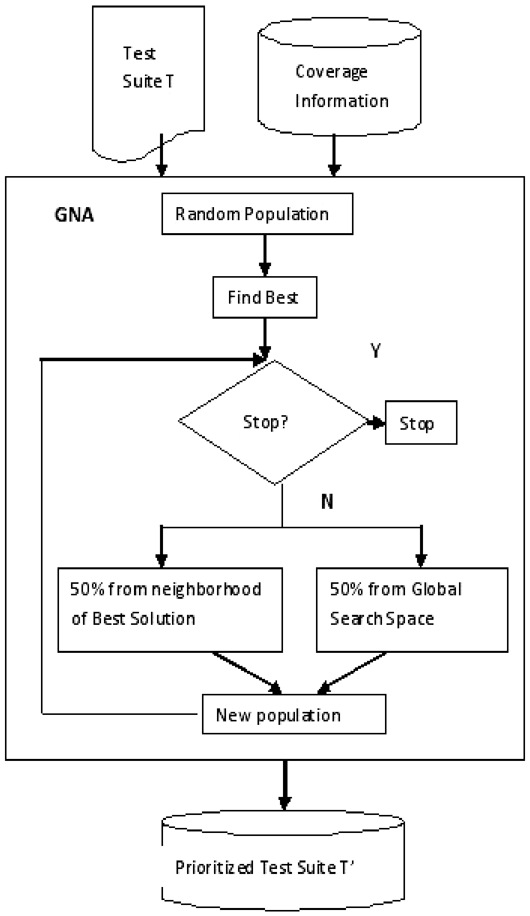
Test Suite T
Coverage Information
GNA
Random Population
Find Best
Y
Stop?
Stop
N
50% from neighborhood of Best Solution
50% from Global Search Space
New population
Prioritized Test Suite T'

Fig. 1: Framework for GNA based TCP

chromosome represents test case numbers. In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence.

- Chromosome A 1 5 3 2 6 4 7 9 8
- Chromosome B 8 5 6 7 2 3 1 4 9

Population: Population is a set of solutions represented by chromosomes. Solutions from one population are taken and used to form a new population. New solutions are selected according to their fitness. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Fitness function: It interprets the chromosome in terms of physical representation and evaluates its fitness based on traits of being desired in the solution. A total of 100% fault coverage in minimum time is taken as fitness criteria. Given Test cases along with the set of faults covered by them and execution time, Fitness function selects test cases and sum up their execution time until all the faults are covered.

Mutation: The mutation operator performs random changes in a chromosome. By doing so, it ensures that new parts of the search space are reached also ensures that no important features are prematurely lost.

Order Changing is the commonly used mutation operator in case of permutation encoding. It is performed by picking two alleles at random and swaps their positions. It Preserves most of adjacency and disrupts the order more.

Example: Consider the following chromosome:

1	2	3	4	5	6	7	8	9

It 2nd and 5th are randomly chosen then after mutation it yields:

1	5	3	4	2	6	7	8	9

Stopping criteria: The Procedure stops when no of generation reaches 10000 or the fitness reaches the limit specified by the tester.

MATERIALS AND METHODS

- Initially ‘m’ Chromosomes which represent different test cases sequences are generated from global search space to form a population
- Evaluate Fitness for each Chromosome in initial population
- Among them, S which has the highest Fitness is chosen and set as current best solution B
- Form new population as follows
  - Generate m/2 Chromosomes from neighborhood of Current Best Solution
  - This done by applying Mutation
  - Generate m/2 Chromosomes from Global Search Space
- Evaluate Fitness for each Chromosome in new population
- Chromosome S with highest fitness is identified and compared with current Best Solution B
- if S better than B then make it the current Best Solution
- Check Stop criterion reached, if so stop otherwise go to step 4

RESULTS AND DISCUSSION

This section presents an experiment set up which includes subject Programs, Techniques involved, metrics, results and their evaluation in order to prove the effectiveness of proposed approach.

Table 1: Subject programs

Test suite	No of test cases	No of statements	No of paths	No of faults
P21	21	50	18	11
P34	34	72	21	15

Test Coverage information needed by Algorithm is considered to be readily available as they can be easily obtained from past runs of Program. It is also out of scope of our research.

Subject programs: Two data sets with test suite sizes of 21 and 34 are taken. The data sets are henceforth referred to as p21 and p34, respectively. Table 1 shows the basic information of these data sets.

Run test case prioritization techniques: Test Case Prioritization algorithms like GA and GNA techniques considered in our experiment. Each algorithm was executed 10 times for the two data sets mentioned in Table 1. About 25% of the execution time of the entire test suite is chosen to represent the tight time schedule. About 50% of the execution time of the entire test suite is chosen to represent for average time schedule. These time budgets are used to prove that the proposed GNA based TCP is more effective even in case of time constraints.

The parameters for the GNA algorithm were as the following:

- Population Size m: 10
- Number of Iterations: 10000

The parameters for the genetic algorithm were as the following:

- Generation size: 10
- Crossover probability: 75%
- Mutation probability: 75%
- Number of iterations: 10000

The bunch of varying results for the same are plotted in Fig. 2-9. Output value/Fitness in above graphs plotted represents the time in seconds needed to achieve 100% fault coverage by the GA and GNA algorithms. By analyzing the graphs, it is found that it produces optimal results over GA almost in all executions for both subject Programs P21 and P34.

Evaluate results: This study presents a graph that compares the proposed GNA based TCP to GA based test case prioritization techniques based on Optimal Results Produced by them and computation Time of them to achieve the same. Three Cases are considered for evaluation:

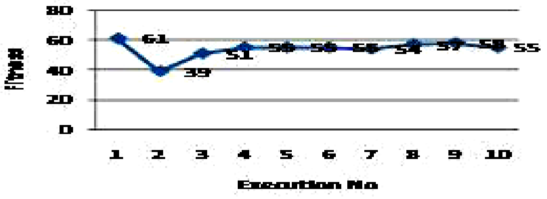

Fig. 2: Execution of P21 in tight time schedule

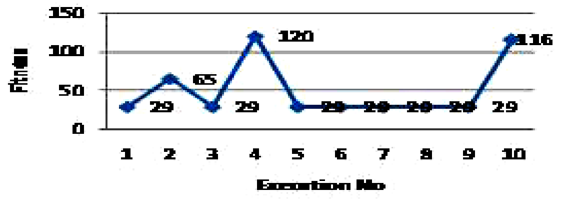

Fig. 3: Execution of P21 in tight time schedule

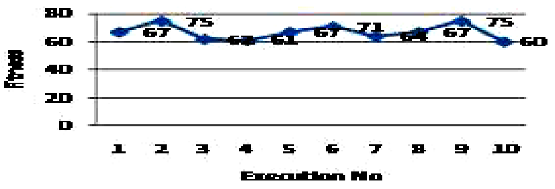

Fig. 4: Execution of P34 in tight time schedule

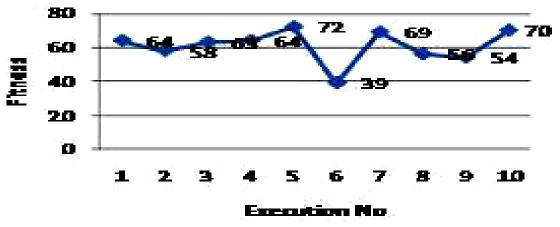

Fig. 5: Execution of P34 in tight time schedule

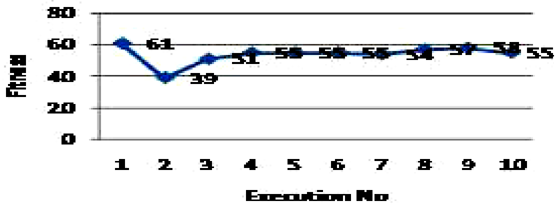

Fig. 6: Execution of P21 in average time schedule

- Best Case-Best among the 10 runs of Algorithm
- Worst-Case-Worst among the 10 runs of Algorithm
- Average Case-average of values of the 10 runs of Algorithm to show the consistency of Algorithm in producing the result

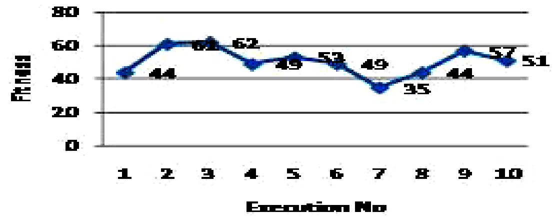

Fig. 7: Execution of P21 in average time schedule

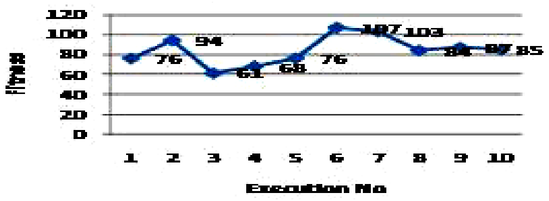

Fig. 8: Execution of P34 with 50% test suite time

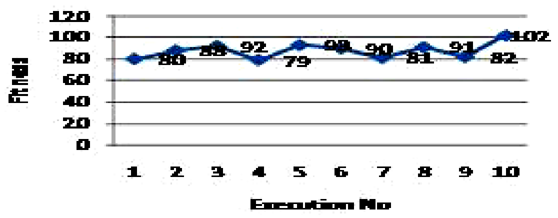

Fig. 9: Execution of P34 in average time schedule

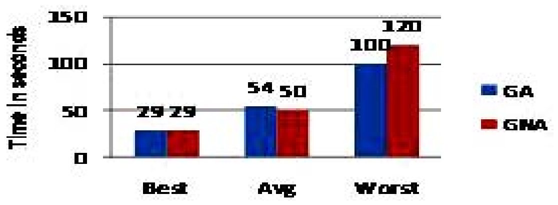

Fig. 10: Execution of P21 in Tight time schedule

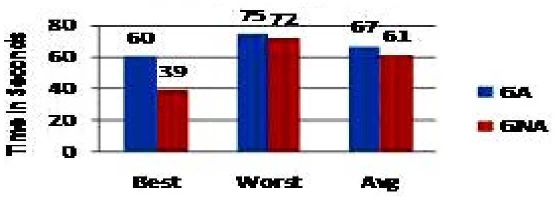

Fig. 11: Execution of P34 in tight time schedule

Figure 10-13 shows that GNA based test cases shows better performance when compared to GA based ones. GNA based Test Case Prioritization performs well and generates global or nearly global optimum solutions in all cases both for tight and average time schedule.

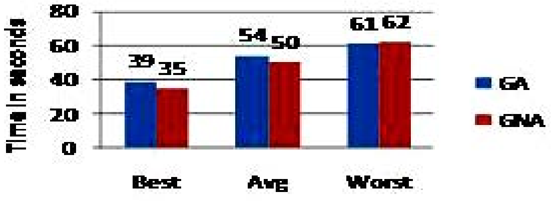

Fig. 12: Execution of P21 in average time schedule

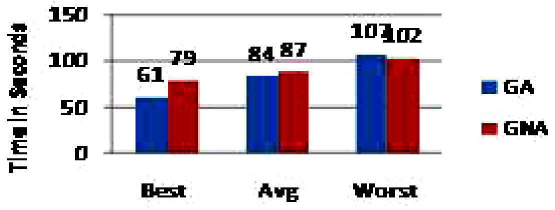

Fig. 13: Execution of P34 in Average time schedule

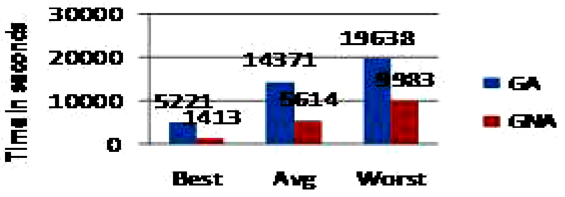

Fig. 14: Execution of P21 in tight time schedule

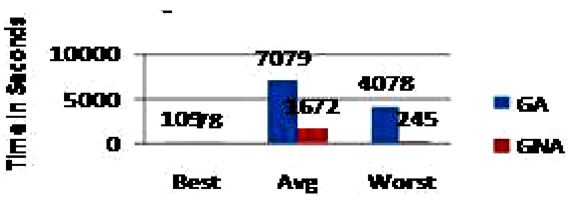

Fig. 15: Execution of P34 in tight time schedule

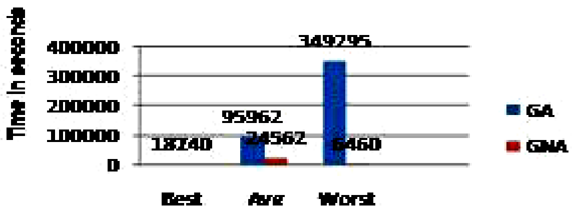

Fig. 16: Execution of P21 in average time schedule

Figure 14-17 shows that computational time of algorithm is very negligible for GNA over GA. It shows that the GNA quickly converge to results in huge search space by transit between Global and Local Searches. The performance of GNA is found to be superior and takes only less time for test case Prioritization process.

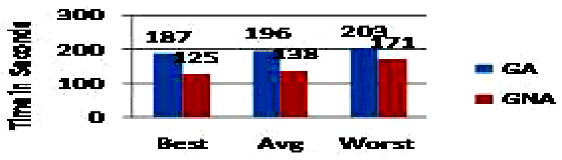

Fig. 17: Execution of P34 in average time schedule

CONCLUSION

TCP strategies allow testers sort the test cases as indicated by certain measure, for example, increasing the rate of fault detection or maximizing code coverage, so that vital test cases are executed before in the testing process. In this study, we exhibited the application of GNA in software test case Prioritization and proved the dominance of the proposed approach over the existing GA based approach. Drawbacks of GA include risk of suboptimal solution and delayed convergence. No assurance for global optimal solution even when it may be attained. But GNA based test case Prioritization produces global or near global optimal results in short time compare with GA. The steps in the proposed approach are very clear and can be effortlessly trailed by software testers. It will surely diminish the time & cost of programming testing and improves the testing procedure and reduces the number of test cases to be examined.

RECOMMENDATIONS

In future, the GNA based approach will be investigated for multi criteria test case optimization and fitness evaluation.

REFERENCES

Alazzam, A. and H.W. Lewis, 2013. A new optimization algorithm for combinatorial problems. Intl. J. Adv. Res. Artif. Intelligence, 2: 63-68.

Jacob, T.P. and T. Ravi, 2013. Optimization of test cases by prioritization. J. Comput. Sci., 9: 972-980.

Krishnamoorthi, R. and S.S.A. Mary, 2009. Factor oriented requirement coverage based system test case prioritization of new and regression test cases. Inf. Software Technol., 51: 799-808.

Maheswari, R.U. and D.J. Mala, 2013. A novel approach for test case prioritization. Proceeding of the International IEEE Conference on Computational Intelligence and Computing Research, December 26-28, 2013, IEEE, Enathi, India, ISBN: 978-1-4799-1594-1, pp: 1-5.

Maheswari, R.U. and D.J. Mala, 2015. Combined genetic and simulated annealing approach for test case prioritization. Indian J. Sci. Technol., 8: 1-5.

Mala, D.J. and V. Mohan, 2009. ABC Tester-Artificial bee colony based software test suite optimization approach. Intl. J. Software Eng., 2: 15-43.

Nagar, R., A. Kumar, G.P. Singh and S. Kumar, 2015. Test case selection and prioritization using cuckoos search algorithm. Proceeding of the International IEEE. Conference on Futuristic Trends on Computational Analysis and Knowledge Managemen, Feburary 25-27, 2015, IEEE, Noida, India, ISBN: 978-1-4799-8432-9, pp: 283-288.

Panigrahi, C.R. and R. Mall, 2013. An approach to prioritize the regression test cases of object-oriented programs. CSI. Trans. ICT., 1: 159-173.

Suri, B., I. Mangal and V. Srivastava, 2011. Regression test suite reduction using an hybrid technique based on BCO and genetic algorithm. Spec. Issue Intl. J. Comput. Sci. Inf., 1: 2231-5292.

Wang, S., S. Ali, A. Gotlieb and M. Liaaen, 2015. Automated product line test case selection: industrial case study and controlled experiment. Software Syst Model., 28: 1-25.

Yu, Y.T. and M.F. Lau, 2012. Fault-based test suite prioritization for specification-based testing. Inf. Software Technol., 54: 179-202.