International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-1, June 2013

Regression Test Selection Using Metaheuristics

Tanvi Agrawal, Arun P. Agrawal

***Abstract*— Regression Testing is a very expensive activity which is to be completed in a very limited time span. Regression test case selection is an effective technique which helps in reducing the cost and time of the testing. To select the efficient test cases for regression test case selection technique, metaheuristic algorithms Tabu Search and Genetic Algorithm are used.**

***Keywords*— Genetic Algorithm, Metaheuristics, NP-hard, Regression Testing, Tabu Search.**

I. INTRODUCTION

Software testing is an important phase of the software development lifecycle. After the software is tested by the testers, the product is given to the customer. The maintenance phase of the software then starts. It is estimated that software maintenance activities approximately account for two-thirds of the cost of the software production [10, 36]. Maintenance of a software product is frequently necessitated to fix defects, to add, enhance or adapt existing functionalities, or to port it to different environments. Regression testing is the testing done during the maintenance phase of a system, component, or a group of related units to ensure that the modification done in the system is working correctly and is not damaging or imposing other modules to produce unexpected results. It is the process of validating modified software to detect whether new errors have been introduced into previously tested code and to provide confidence that modifications are correct. Since regression testing is an expensive process, several methods have been developed in an attempt to reduce the cost of regression testing. An important research problem is the selection of a significant subset of test cases from the initial test suite that would minimalize both the effort and regression testing time without sacrificing the meticulousness of regression testing [2]. Test selection techniques normally use the source code of a program to determine which tests should be executed during the regression testing stage [4].
The series of processes that take place after the release of the software in the maintenance phase is shown in Figure 1 [2]. The different types of regression testing are: 1) Retest All 2) Regression Test Case Prioritization and 3) Regression Test Case Selection (RTS). Regression test case selection techniques select a subset of valid test cases from the previous test suite to test that the affected but unmodified parts of a program continue to work correctly [7]. Use of an effective regression test case selection technique helps in reducing the testing costs in environments in which a program undergoes frequent modifications [5]. Regression test selection essentially consists of two major activities:
– Identification of the affected parts - This involves identification of the unmodified parts of the program which are affected by the modifications done in the program.
– Test case selection - This involves identification of a subset of test cases from the initial test suite which can effectively test the unmodified parts of the program. The aim is to select the subset of test cases from the initial test suite that has the potential to detect errors which were generated due to the changes in the program.

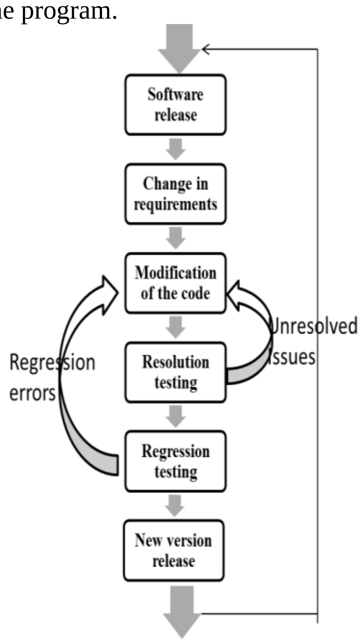

Figure 1: Processes in software maintenance and regression testing

II. METAHEURISTIC

Many practical problems are NP-hard problems in nature, which means complete, constructive search is unlikely to satisfy the computational demand [8]. Generally, optimization problems are NP-hard, complex and time consuming. There are two ways to tackle these types of problems: **exact methods and approximate algorithms**.

A. *Exact Method*

It allows finding exact solutions but is often impractical as they are very time-consuming for real-world problems. The classical algorithms such as branch and X family of algorithms, A* family of search algorithms, dynamic programming and constraint programming comes under the class of exact methods [9]. Optimal solutions are obtained from exact methods and their optimality is guaranteed.

B. *Approximate Algorithm*

There are two subclasses of algorithms under the class of approximate algorithms i.e. *approximation algorithms and heuristic algorithms*.

Manuscript published on 30 June 2013.
***Correspondence Author(s)**
Tanvi Agrawal, CS&E Department, Amity University, Noida, India.
Arun P. Agrawal, CS&E Department, Amity University, Noida, India.

In approximation algorithms, there is a guarantee on the bound of the obtained solution from the global optimum [9, 380]. Heuristics are approximate solution techniques which have been used in the operations research to tackle difficult combinatorial problems. With the development of complexity theory, it was clear that, since most of these problems were *NP-hard problems*, there was less chance of ever finding efficient exact solution procedures for them. This realization highlighted the role of heuristics for solving the combinatorial problems that were faced in real-life applications and needed to be tackled, whether they were *NP-hard problem* or not. Optimization problems are generally NP-hard and heuristic solution methods are needed to solve these optimization problems. Heuristics are classified as: *specific heuristics* and *metaheuristics* [9]. Specific heuristics solves a specific problem and/or instance of any problem. Metaheuristics algorithms can be used to solve almost any optimization problem. **METAHEURISTIC:** 'Meta' means *abstract*, and a 'heuristic' is a *search*. A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions [6]. It is a strategy that "guides" the search process. Metaheuristics provide sub-optimal (sometimes optimal) solutions in a reasonable time. Thus, metaheuristics usually allow meeting the resolution delays imposed in the industrial field as well as they allow to study general problem classes instead that particular problem instances. In general, many of the best performing techniques in precision and effort to solve complex and real-world problems are metaheuristics. Their fields of application range from combinatorial optimization, bioinformatics, and telecommunications to economics, software engineering, etc. The main aim of metaheuristics is to efficiently explore the search space in order to find (near-) optimal solution. Techniques which have metaheuristic algorithms range from simple local search processes to complex learning procedures. Metaheuristic algorithms are usually approximate and non-deterministic. For complicated search problems and NP-hard optimization problems, metaheuristic methods are better choices for solving these problems. These are designed to solve complex optimization problems where other optimization methods have failed to find effective and efficient solution. These methods have been recognized as one of the most useful methodologies for solving many complex problems, and particularly many real-world problems that are combinatorial in nature. Metaheuristics can also be classified as *deterministic and stochastic metaheuristics* [9]. The deterministic metaheuristic makes deterministic decisions to solve the optimization problems and also the same initial solution leads to same final solution i.e. there is no change in the final result (e.g. tabu search). In the stochastic metaheuristic, some random rules are applied during the search process. Also the same initial result does not produce same final solution instead different final solutions are obtained. Hence, this characteristic must be remembered while performing any evaluation of metaheuristic algorithm.

Various types of metaheuristic algorithms are Genetic Algorithm, Tabu Search, Simulated Annealing, Variable Neighborhood Search, Iterated Local Search, Ant Colony Optimization, Evolutionary Computation, Particle Swarm Optimization, Artificial Bee Colony and GRASP.

C. Tabu Search (TS)

Local Search (LS) gave an idea that a known solution S can be improved by applying small changes. The solution obtained after the modification of the solution S is known as neighbors of S. LS begins with some initial solutions and travels from one neighbor to another as much as possible and keeps on decreasing the function value. The major issue with this approach is that it cannot escape from local minima and the search is not able to find any more neighborhood solutions which would decrease the objective function value. One of the best methods to solve this problem is Tabu Search (TS). Tabu Search permits LS approach to overcome local optima and to implement an explorative strategy [11]. The simple methodology of TS is to follow LS whenever it meets local minima by permitting non-improving moves cycling back to previously visited solutions is prevented by the use of memories, called tabu lists, which record the recent history of the search, a key idea that can be linked to Artificial Intelligence concept [1]. Tabu Search is a metaheuristic proposed by Fred Glover in 1989 which helps in solving combinatorial optimization problems. TS is not like other metaheuristics. It is based on the idea that an intelligent search has to perform a systematic exploration of the solution space [12]. TS is a solution-to-solution based method. The tabu list L_k is dynamic and after each move, the latest solution $\theta_{k,}$ or the move that resulted in this solution, is added to the list and the oldest solution or move is removed from the list [3]. Another characteristic of TS is that it always selects the finest solution from the neighborhood which is not on the tabu list, even if it is not better than the present solution. This allows the search to overcome local optima, and tabu list confirms that the search does not come back.

D. Genetic Algorithm (GA)

The original inspiration for the GA methodology was a biological analogy. In the selective breeding of plants or animals, for example, offspring are sought that have certain desirable characteristics—characteristics that are determined at the genetic level by the way the parents' chromosomes combine [1]. A population of string known as chromosomes is used in GAs. Analogies such as genetic crossover and mutation are used in recombination of strings. The search is guided by the results of evaluating the objective function f for each string in the population [1]. Based on this evaluation, strings that have higher fitness (i.e., represent better solutions) can be identified, and these are given more opportunity to breed [1]. . It tries combining various solutions to evolve the better ones. It also uses mutation to ensure that maximum numbers of solutions are explored, as fast and efficiently as possible. For evolutionary strategies, search is entirely based on mutation of solutions whereas GA uses recombination [13]. Both the techniques can be combined also and have a very strong analogy with evolution in nature [13, 15]. A collection of individuals also called population is used in producing offspring by combining genetic material. An evolutionary or genetic algorithm develops the similar system to evolve solutions.

Evolution is based on the fitness of an individual, just as in biology; this fitness is calculated by a fitness function [13]. This search algorithm is a set-based algorithm and generates a number of solutions at each iteration. For all iterations, a subset of the solutions is selected based on their performance from the current set of solutions. These solutions are further combined to form new solutions. GA is quite robust in correspondence to which solutions must be selected in order to create the next set.

III. PROBLEM DESCRIPTION

RTS is an optimization problem as it has to fulfill two goals: (i) Re-test the unmodified parts of the program on the basis of test requirements and (ii) Reduce the cost of regression testing. RTS is a NP-complete problem because it cannot give a solution in the polynomial time.

There is a problem P for which 'n' test cases is given in the test suite T with their execution time t_i. The problem is modified as P' and is to be tested again. A new test case suite T' with m test cases is to be selected using RTS from the previous test suite T. The execution time in T' is t_j'.

Constraints

(i) Total Execution Time $\sum_{j=1}^{m} tj \lesssim \sum_{i=1}^{n} ti$, where m$\lesssim$ n.

(ii) Code Coverage of T' ≍ T.

(iii) No. of faults discovered in T' ≍ T.

IV. EXPERIMENTS AND RESULTS

The experiment was done using Heuristic lab for both Tabu Search and Genetic Algorithm. They were run for Travelling Salesman Problem (TSP). TSP is a combinatorial optimization problem and hence can be classified as NP-complete problem as proved by many researchers. This test was conducted with the following parameters for the GA and TS:

Parameters for GA:

- Population Size =1000
- Mutation Probability = 10%
- Maximum Generations= (See table below)

Parameters for TS:

- Neighborhood Size =1000
- Tabu Tenure =10
- Maximum Iterations = (See table below)

Since both RTS and TSP are NP- complete problems. Therefore, parameters for Test Case Selection

- No. of cities in TSP = No. of Test Cases generated
- Distance between two cities = Execution Time of Test case

Iterations	Tabu Search (Best Quality)	Genetic Algorithm (Best Quality)
100	717	2414
200	651	1608
300	668	1373
400	652	698
500	660	669
600	656	701
700	663	675
800	647	692
900	663	663
1000	651	688
Average	662.8	1018.1

Table 1: Best Quality achieved at different iterations where Best known Quality = 629

In the first experiment, the aim was to find the best quality for the problem (TSP). The best known qulaity was given and the best quality was obtained for both the algorithms. The result is shown in Table 1. Also, Table 2 shows the absolute difference for both the algorithms.

Iterations	Tabu Search (Absolute Difference)	Genetic Algorithm (Absolute Difference)
100	88	1785
200	22	979
300	39	744
400	23	69
500	31	40
600	27	72
700	34	46
800	18	63
900	34	34
1000	22	59
Average	33.8	389.1

Table 2: Absolute Difference between Best known Quality and Best Quality obtained

The second experiment was conducted to comapre the execution time for both the algorithms for the same parameters. This was done by comapring the time taken to generate the best quality with the above maximum iterations. Table 3 gives the execution time for both the algorithms with the average time. Figure 2 and figure 3 shows the solution obtained by both the algorithms TS and GA respectively.

Iterations	Tabu Search (Time)ms	Genetic Algorithm (Time)ms
100	4300	9300
200	8500	18500
300	12700	27400
400	16900	36500
500	21300	45700
600	25500	55300
700	29700	63900
800	33900	73100
900	38900	8300
1000	4220	91100
Average	19592	42910

Table 3: Time taken o execute the problem.

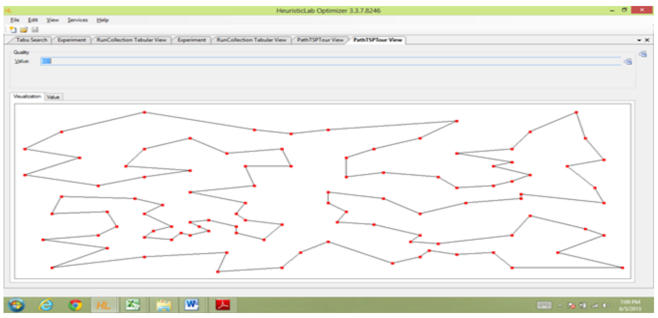

Figure 2: Solution of TSP using Tabu Search

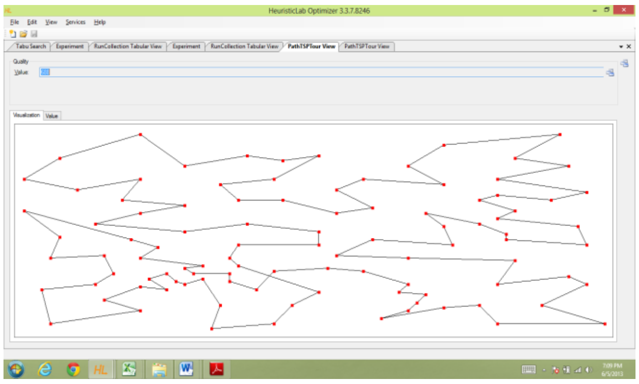
HeuristicLab Optimizer 3.3.7.8246
File Edit View Services Help
Tabu Search
Experiment
RunCollection Tabular View
PathTSPTour View
Quality
Value
Visualization
Value

Figure 3: Solution of TSP using Genetic Algorithm

V. CONCLUSION

It has been observed that both the algorithms TS and GA has been executed on TSP generate the best quality result in a definite execution time. The overall quality of the problem is evaluated by the absolute difference in best quality that adds up violation of all constraints. The results are shown in tables above. Table 1 compares the best quality generated by both the algorithms and table 2 compares the absolute difference between best known quality and best quality. It gives the average best quality of TS and GA as 662.8 and 1018.1respectively. Table 3 shows that the average time of executing the problem with the above quality is 19592 ms and 42910 ms respectively. The obtained results show that TS is a better algorithm for Regression Test Case Selection than GA as it takes less execution time and gives a better quality value.

REFRENCES

1. F. Glover and G. A. Kochenberger. (2003). Handbook of Metaheuristics [Online]. Available: http://kluweronline.com.
2. S. Biswas and R. Mall, “Regression Test Selection Techniques: A Survey”, Informatica 35, 2011.
3. S. Ólafsson,”Handbook on Simulation: Metaheuristics”, Nelson and Henderson, p. 633-654.
4. G. M. Kapfhammer, “Software Testing”, Department of Computer Science Allegheny College.
5. T. L. Graves, M. J. Harrold, JM Kim, A Porter, G Rothermel, “An Empirical Study of Regression Test Selection Techniques”, ACM Transactions on Software Engineering and Methodology, vol. 10, no. 2, April 2001, P. 184–208.
6. C. L. B. Maia, R. A.F. Carmo and F. G. Freitas “A Multi-Objective Approach for the Regression Test Case Selection Problem”.
7. S.Sundar and A. Singh, “A hybrid heuristic for the set covering problem”, September 2010, Springer.
8. C. L. B. Maia, R. A.F. Carmo and F. G. Freitas “A Multi-Objective Approach for the Regression Test Case Selection Problem”.
9. E.G. Talbi. (2009).Metaheuristics from design to implementation. [Online].Available: www.wiley.com.
10. G. Rothermel and M.J. Harrold, “Analyzing Regression Test Selection Techniques”, IEEE Transactions on Software Engineering, vol. 22, no. 8, August1996.
11. C. Blum and A. Roli, “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”, ACM Computing Surveys, Vol. 35, No. 3, Sept- 2003, p. 268–308.
12. R. A.Valdés, E. Crespo and J. M. Tamarit, “Tabu Search: An efficient Metaheuristic for University Organization Problems”, Revista Investigation Operacional, Vol.22, No.2,2001.
13. J. Oenen, “Improving Regression Test Code Coverage with Meta-heuristics”, M.S dissertation, Dept. Computer Science, Delft University of Technology, 2008.

AUTHOR PROFILE

Tanvi Agrawal She has obtained her B.E. degree in Information Technology from Rajasthan University. Presently she is pursuing her M.Tech in Computer Science and Engineering from Amity University, Noida. She has 8 months of teaching experience.

.

Arun Prakash Agrawal He has ten years of experience of teaching at graduate and post-graduate levels. He obtained his M.Tech. in Computer Science and Engineering from Guru Gobind Singh Indraprastha University, Delhi. He is also the gold medalist of his batch. He has taught at various engineering colleges of repute. Presently he is pursuing his Ph.D. in the area of software engineering from Guru Gobind Singh Indraprastha University, Delhi. He has several research papers of national and international repute to his credit. He He is a member of IEEE computer society, IAENG and ACM. His area of interest includes software engineering, software testing, computer networks and mobile computing.