Adaptation Of Tabu Search And Cognizant Turing Machine For Regression Test Case Prioritization

V.Ranjani, K.Jeya Aishwarya, A.Jeya Vani, S.Sangeetha

Guided by, P.Sundaravadivel (*Assistant professor)*
Department of Computer Science, SBM college of Engineering and Technology, Dindigul.

Abstract:

Regression testing is highly-priced, but important thought-process-action in software testing. Computational complexity issues are gaining increasing attention as test case prioritization techniques mature. Test case prioritization techniques rearrange test cases to increase the rate of fault detection. The objective of the study is to have an in-depth investigation about regression test suite prioritization to improve the performance measures. In this paper we propose a new test case prioritization technique based on integrating tabu search and turing machine to solve the statement coverage and fault detection problem. This study analyzes integrating the tabu search and turing machine cognizant with regard to computational overhead by utilizing hierarchical order-based criterion to prioritize test cases. Tabu search is a searching tool to find near-optimal solution. Thought-Process-Action Turing machine is used to verify and validate that all the faults are covered by the test cases picked in the prioritized queue. The proposed technique give rise to the ordering of the original test suite so that the new suite, which run within a time-limit execution will have a increasing rate of fault detection when compared to rates of randomly prioritized test suites.

Keywords: Machine cognizant; Regression Testing; Test case Prioritization; Thought-Process-Action Turing Machine [CTM].

1. Introduction

Tabu Search is a powerful optimization procedure that has been successfully applied to a number of combinatorial optimization problems [1-6]. It has the ability to avoid entrapment in local minima by employing a flexible memory system. In [7], a simple TS algorithm based on short-term memory has been proposed for solving the unit commitment problem. Specific attention is given here to the short-term memory component of TS, which has been considered as the simplest form of TS procedures [7]. Rothermel formally defined the test case prioritization problem and empirically investigated six priorization techniques. Four of the techniques were based on either statements or branch coverage. The remaining two techniques were based on the estimated ability to identify and detect faults [10]. Optimal solution is searched for a given problem based on the evaluation of objective function. In previous studies, results show that most of the existing techniques could improve the rate of faults detection, but they have different performance and complexity in various situations. This study analyzes the tabu search with regard to effectiveness and time overhead by utilizing hierarchical order based criterion to prioritize test cases. The organized and methodological use of tabu search is an crucial feature of Tabu search. Tabu search belongs to the class of dynamic adjoining search techniques. In this paper, two most popular measure are taken in to consideration: statement coverage and fault detection are considered for test case prioritization. TABU search is employed as the adaptive selection methods for test suite prioritization. Alan Turing describes the "Turing test," an imitation game wherein the intelligence of the machine is tested via its ability to sustain human-like discourse. Turing mathematically proved the Church-Turing Thesis [12] [13], stating that, any computational task that can be performed by any physical system whatsoever, such as the human brain, can be performed by a relatively simple digital computer called a Turing Machine. Basically, we introduce a set of turing machine and Tabu search used for expansion and transformation together with the CTM is used for fault detection. The new parameters of the tabu search are automatically adjusted and the input symbols of the turing machine represent additional items to be arbitrarily determined by

the Tabu search algorithm or by the user based on certain constraints. The Proposed technique will be improved with the interaction of the machine and man-kind. Happily, the machine can think, process the input and act resembling human with important limitations. We may improve the performance, by the secondary approach, turing machine should be considered in the training process of prioritization. The execution order of the test cases was determined by the class of the test suite to be executed, the execution order is predetermined, except possibly for opportunistic optimization.

2. Proposed System

The objective of test case prioritization is to increase the rate of fault detection by reordering the test case with minimum execution time. The main problem with tabu search is to get away from local minima where the objective value cannot be decreased further. Thought Process Action Turing Machine is used to verify that all the faults are covered by the test cases selected in the prioritized queue by feed-back approach. In this method, the program code is modified, then recompile and run a suite of valid test cases against the changed program.

2.1. *Tabu Search*

The basic idea of Tabu Search (TS) was introduced by Fred Glover [20]. The basic concept of TS as described by Glover [6] is "a metaheuristic super imposed on another heuristic ". TS is a meta heuristic search capable of carrying the search process beyond the universal optimum. We proposed a technique that prioritizes regression test suites so that the new ordering will always run within a given time limit and will have high prospective for defect detection based on resultant coverage information. The pathway to the universal optimum is guided by the combination of adjoin searching and recording the past solution, which is maintained by the memory.

The TS is basically used as an exploration tool. The exploration process is made easy by a procedure called move. The recently visited solution are removed from the search space and consider as tabu solutions. The move defines the next solution to be visited in the search space. TS examines the adjoin searching of current solution and it selects the best among all based on the objective function. The memory transposition and current solutions are used to generate the members of an adjoining list. In TS, an optimal solution along with the path, which is used for backtracking, plays an important role. The pathway to the universal optimum is derived from the adaptive memory, which becomes the feature of TS. The important objects of the TS are expansion and transformation. The expansion is based on the concept that 'a good solution may be optimal or it may lead to the optimal solution. It involves the thorough inspection of solution and it's adjoining list. This process is used to reach the optimal solution by fixing the direction of next move. The standstill at the universal optimum becomes the drawback of expansion and it is overcome by the transformation. The transformation process examines the unvisited solution so far and it may show the way for global optimum. The tabu list is used to implement expansion and transformation. Turing machine is also used for transformation. Transformation will thus trend to spread the exploration effort over different test cases of the test suite. The transformation phase is directed towards unexplored test cases.

2.2. *Turing Machine cognizant or Thought Process Action TM*

In the early period of computer development, Alan Turing focused on human mechanical (such as mechanical/electronic computers) calculability with symbolic configurations [12]. Mathematicians view the Church-Turing Thesis as an expansion of the capability of digital computers to solve mathematical and symbolic functions with a precision equal to or greater than the capability of any other mathematical solving system (such as a slide rule or the human brain) [12] [14]. The Church-Turing Thesis is that effectively calculable functions of positive integers (signals commonly used in modern digital computers and in the Turing Machine) should be identified with that of a mathematically precise recursive function. The Church-Turing hypothesis proves that a digital computer (operating on mathematical digital signals) may be used to solve all functions just as effectively, and with the same mathematical precision, as the non-computer like methods. Problems that can be solved in principle may not be solvable in practice using logic. We construct a CTM to make the set of rules less tedious. All test suites that are taken into account for preplanning. The execution order of test case suite is pre planned and goal directed with the option available for

replanning the prepared path if a time limit contingency is detected prior to reaching the preplanned goal. Re-planning is always a function of the contingency that appears in the area of the pre-planned path. Turing applied the

computational capability of computers to human "thinking" and cognition [13].In this paper we shall assume the same as turing did , that "Cognitive thinking is a subjective experience". The question avoided by Turing, through the invocation of a argument, is: What are the competence conditions required to both perform symbolic computation and also generate the perceptive knowledge, that are practiced by all human observers?. The difference between human mind and a turing machine is the set of variables that frame the input and output states of the system. The turing machine can protract human like discourse. CTM is used to verify that all the faults are covered by the test cases selected in the priority queue. The program code is changed , then re-run a suite of valid test cases against the changed program. The TS-set Domain of Digital Computers Programmed by Tabu search algorithmic programming. The CTM set is the set of perceptive experience, understood by the human mind. The members of the CTM-set do not follow the rules of TS-set- symbolic mathematics. The Symbol between these two set must adhere to the computational requirements of the CTM-set and not the TS-set. CTM is centered on the existence of TS-set coding schemes that transforms the members of TS-set into members of CTM-Set, CTM=F(T). The new test suite is as same as the parent suite apart from one or more changes may be made to the new test suite test cases. The computational task performed on the (CTM+TS) –set in F(n) steps, where n is the size of the input , can be performed by a turing machine in G(n) steps, where F,G differ by at most a polynomial [11]. By not recognizing the independence of the "cognitive thinking" variable in the CTM-set, Turing, most likely, discouraged the search for a sufficiency condition applied to "human discourse" and "conscious thinking." All the programming of the machine is performed with a finite, non-exponential number of steps.CTM machine may experience the subjective measures of visualization or thought that the test cases it interacts with. Machine learning is needed so that computers can discover new knowledge inductively from experience as well as deductively. Human interactions are required to make machines think and act like humans. Thus, if a machine can sustain humanlike discourse, and it is programmed to "pretend" that it experiences "cognitive thinking," then it is impossible to prove otherwise [16]. And, "instead of arguing continuously over the point," Turing himself

states that "it is usual to have the polite convention that everyone thinks" (both humans and the conversing Turing Machine) [13]. Thus, Turing was not able to prove that if a Turing Machine can sustain human-like dissertation then it has a human-like thinking capability. But he stated it as a hypothesis. It is thus possible show that a cognizant Turing Machine exists, if it has the capability to transform members of the CTM-set into members of the TS-set. Then they are subject to transposition with a small user specified input symbol and value V. If a random number R is selected between [0,1] is less than the user specified value V for the test case CTM, new test not included in the current test suite is randomly selected from T to replace CTM. Instead of replacing the test with a random test, the test to be transformed is swapped with the test cases that succeeds it. The modified objective transition function may in some instances be replaced by another function . The process of making turing machine cognizant may include a new objective function§ .

§= [TS+CTM]+ Expansion + Transformation

Thus we are capable to show that the cognizant turing machine exists under certain constraints. The new test suite is same to the prior parent suite excluding one or more changes to the test cases of the new test suite.

2.3. *Process flow of the Proposed Technique*

Step 1: Collate the test cases. Collate the test cases elicited during the testing, and give the stakeholders the chance to contribute new ones. Prioritize these test cases based on tabu search satisfying the business goals and choose the top one-third for further testing.

Step 2: Refine test cases: Refine the test cases output from step 1, focusing on their stimulus response measures. Elicit the worst case, current, desired, and best case quality attribute response level for each test suite.

Step 3: Prioritize test cases. Assign a weight of 1.0 to the highest rated test cases; assign the other test cases a weight relative to the highest rated. Make a list of quality attributes that concern the stakeholders.

Step 4: Assign utility. The utility for each test suite is assigned possibly on timebudget. A utility score of 0 represented no utility. A score greater than 0 represented the utility possible execution.
Step 5: Develop the Cognizant Turing Machine strategy for each test suite and determine their faults identified and its execution time levels. Develop the turing machine strategy that address the chosen test suite and determine the fault.
We must perform cognizant turing machine computation to calculate the total number of faults. We use CTM to detect more faults earlier in the execution of test cases. Intelligent prioritization can be done by CTM. Reconsidering the individual exploration strategy .
Step 6: Determine the utility of the "expected" quality attribute response levels by interpolation [Machine-Man Kind]. Determine the average number of faults detected per unit time. Do this for each relevant quality attribute enumerated in step 3.
Step 7: Calculate the total benefit obtained from the combined strategy. Subtract the utility value of the current level from the expected level and normalize it using the weight scheme elicited in step 3. Sum the benefit due to a individual strategy across all test cases and across all relevant quality attributes such as time, error rate,etc.
Step 8: Choose combined strategies based on schedule constraints. Determine the cost and schedule implications of each test cases. Rank-order the test cases according to the time constraint and choose the top ones until the time budget or schedule is exhausted. Consider the number of fault identified within the time-limit.
Step 9: Confirm the results with intuition [20]. For the chosen individual strategy, consider whether these seem to align with the time based constraint ordering of test cases. If there are significant issues, perform another iteration of these steps to reorder the test cases.

3. Conclusion and Future Work

This technique has been proposed to prioritize test cases using Tabu search and turing machine. Integration in this area is still incomplete. The ordering of test cases can exploit only relatively common test case information. Steps toward more complexity show up in the integration of meta models to describe the sharing, distribution, data merging between these two approaches. The most significant aspect of this design is that the input and output of the proposed technique is incremental, so a component that needs to understand their operation must retain and update a history of the interaction.

Acknowledgments

Our sincere thanks to Mr.James, PS-Controller of examination, Christ the King Engineering College, Karamadai, Coimbatore,Tamilnadu, India and Mr.M.P.Muthusamy, Managing secretary, Temple of consciousness, Cumbum, Theni district,Tamil nadu for their valuable support in framing this document.

References

[1] A. Bland and G. P. Dawson, "Tabu Search and Design Optimization", Vol. 23, No. 3, pp. 195-201, April 1991.
[2] F. Glover, " A User's Guide to Tabu Search ", Annals of Oper. Reas. 41(1993) 3-28
[3] F. Glover, "Artificial Intelligence, Heuristic Frameworks and Tabu Search", Managerial and Decision Economics, Vol.11 ,365-375(1990).
[4] F. Glover , H. J. Greenberg, " New Approach for Heuristic Search: A Bilateral Linkage with Artificial Intelligence", European Journal of Operational Research 39 (1989) 119-130.
[5] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence", Comput. & Ops. Res. Vol. 13, NO. 5, pp. 533-549, 1986.
[6] F. Glover, "Tabu Search-Part I", ORSA Journal on Computing, Vol. l., No. 3, pp. 190-206, 1989
[7] A. H. Mantawy, Youssef L. Abdel-Magid, and Shokri Z.Selim, "Unit Commitment by Tabu Search ", IEEE proceedings - Generation, Transmission and Distribution, Vol. 145, No. 1, January 1998, pp. 56-64
[8] Zheng Li, Mark Harman, and Robert M. Hierons, "Search Algorithms for Regression Test Case Prioritization", IEEE Transactions on software Engineering, vol.33, No.4, April 2007 , pp. 225-237.
[9] Lu Zhang, Shan-Shan Hou, Chao Guo, Tao Xie, Hong Mei: Time-aware Test Case Prioritization Using Integer Linear Programming, Proceedings of the 2009,ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '09), pp. 213-224, 2009.
[10] G. Rothermel, R. J. Untch, and C. Chu, "Prioritizing test cases for regression testing", IEEE Transaction on Software. Eng., 27(10), 2001, pp. 929-948.
[11] Levesque, H (1986) Knowledge representation and reasoning, Annual review of computer science. 1:255-287

[12] Church, A. (1937) Review of Turing (1936). J. Symbolic Logic 2:42-43
[13] Turing AM (1950) Computing Machinery and Intelligence, Mind 59: 433-460
[14] Turing AM (1953) Solvable and Unsolvable Problems. Science News 31: 7-23.
[15] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies", IEEE

[16] Transactions on Software Engineering, 28(2), 2002 , pp.159–182.
[17] Rosen, D.B., & Rosen A, "The Turing Machine Revisited: The computational complexity of a visually conscious machine. Can a conscious machine exist?". (available for viewing at www.mcon.org)
[18] P. Chu and J. Beasley, "A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics", 4(1), 1998 , pp. 63-86.
[19] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: An empirical study", In Proceedings ICSM 1999, Sept. 1999 , pp. 179–188.
[20] R. Krishnamoorthi, S.A. Sahaaya , and A. Mary, "Regression Test Suite Prioritization using Genetic Algorithms", International Journal of Hybrid Information Technology, 2009.
[21] Len Bass, Paul Clements, Rick Kazman. Software Architecture in Practice, Pearson Education, Second Edition, 2004.