Test Case Prioritization for Regression Testing Based on Ant Colony Optimization

Dongdong Gao
Beijing Institute of Control Engineering
Beijing 100190, P.R.China
gaodongdong@buaa.edu.cn

Xiangying Guo and Lei Zhao
Beijing Sunwise Information Technology Co.Ltd.
Beijing 100190, P.R.China
{guoxiangying &zhaolei}@sunwiseinfo.com

Abstract—**Test case prioritization technique is an efficient method to improve regression testing activities. It orders a regression test suite to execute the test cases with higher priority earlier than those with lower priority, and the problem is how to optimize the test case ordering according to some criterion. In this paper, we have proposed an algorithm which prioritizes the test cases based on ant colony optimization (ACO), considering three factors: number of faults detected, execution time and fault severity, and these three factors are used in ant colony optimization algorithm to help to reveal more severe faults at earlier stage of the regression testing process. The effectiveness of the algorithm is demonstrated using the metric named APFD, and the results of experiment show the algorithm optimizes the test case orderings effectively.**

Keywords- test case prioritization; ant colony optimization; regression testing

I. INTRODUCTION

Regression testing is one of the most critical activities of software development and maintenance, and it is performed when changes are made to existing software. The purpose of regression testing is to ensure that no additional errors were introduced in the process of fixing other problems or modifications of software [1]. Regression testing is often an integration testing process, and it involves execution of large number of test cases and is time consuming [2].

To optimize the time and cost spent on regression testing, there are four methodologies used in the process: retest all [3, 4], regression test selection [5, 6], test suite reduction [7, 8] and test case prioritization [9, 10]. Retest all technique is a conventional method for regression testing in which all the tests in the existing test suite are re-runned, and it may be expensive as it takes time and effort. Regression test selection method selects some appropriate subset of the existing test suite to reduce the cost of regression testing, and test suite reduction method reduces the test suite to a minimal subset which is equivalent to the original test suite based on redundancy information [11]. One drawback of regression test selection and test suite reduction is that they might reduce the fault detection capability with the reduction of test suite size [10]. Test case prioritization technique provides another method to reduce regression testing cost, and it schedules test cases in an execution order according to some criterion. In this paper, we focus on test case prioritization technique. There are many algorithms used for test case prioritization in the previous works, and soft computing constitutes a high proportion such as genetic algorithm (GA) or ACO [12]. GA-based approach is applied to prioritize a regression test suite within a time constrained execution environment [13, 14]. Arvinder Kaur et al. [15] proposed a bee colony optimization algorithm for code coverage test suite prioritization, and the Average Percentage Condition Coverage metric is used to show the effectiveness of proposed algorithm. Bharti Suri et al. [16-18] proposed test case selection and prioritization approach based on ACO to find the solution that is nearest to the optimal solution, and two factors, which are fault coverage and execution time, were considered in the algorithm. Lu Zhang et al. [19] studied the use of integer linear programming for test case prioritization, and compared his approach with the GA-based approach and traditional approach. Previous studies on test case prioritization using ACO do not consider different fault severities, and it may provide unsatisfactory results if all the faults are not equally severe. To solve this problem, a new ACO algorithm for test case prioritization is proposed in this paper.

The paper is organized as follows. The next section of this article constructs the formulation of the test case prioritization problem, and discusses three factors to be considered for prioritization. Section III presents the design of our algorithm, and provides a metric method to validate the effectiveness of the algorithm. Case study is conducted, and the performance of the algorithm is analyzed through experiments in Section IV. Finally, Section V gives the conclusion about our work.

II. PROBLEM FORMULATION

A. Test Case Prioritization Problem

In this article, we consider the test case prioritization for regression testing. Assuming P is a program, and P' is a modified version of P. Let T be a test suite developed for P, and a selective regression testing technique is concerned with validating P' by reusing T. The process of reusing T is related to test case selection and prioritization. Test case prioritization techniques schedule T, or a subset of T in an order according to some criterion. Rothermel et al. define the test case prioritization problem as follows [20, 21]:

Given: T, a test suite; PT, the set of permutations of T; f, a function from PT to the real numbers.

978-1-4799-8353-7 /15/$31.00 ©2015 IEEE

This work is being supported by the Natural Science Foundation of China under Grant No. 91118007.

Problem: Find $T' \in PT$ such that $(\forall T'')\,(T'' \in PT)(T'' \neq T')[f(T'') \leq f(T')]$.

The notations used above are explained below, PT is the set of all possible prioritized test case orderings of T, and f is an objective function that maps any order in the set PT to a real number. The function f represents a quantification of the prioritization goal, for simplicity and without loss of generality, it is reasonable to assume that high value of the function f is superior to low value. T' and T'' are the optimal and any test case order in the set PT, respectively.

B. *Factors to be Considered for Prioritization*

Test case selection and prioritization is influenced by many possible factors. Three factors are considered to prioritize the test cases in our proposed algorithm, and these factors are discussed below.

- Number of faults detected by a test case. A test case covers one, or possibly more faults in its execution. Fault coverage capacity of a test case shows the quantity of faults detected by it. The test case covering more faults should be scheduled higher priority in the test case ordering.
- A test case execution time. Time cost of a test case plays a significant role in an efficient regression testing, and the time constraints on regression testing usually does not allow the execution of the entire test suite. Consequently, a test case execution time is considered in test case prioritization.
- Fault severity. Different faults have different impacts on the same software, and the acceptance of the fault is determined by its severity. In order to find more severe faults in the early testing phase, the test case revealing severer faults should be executed earlier.

There is another factor named rate of fault detection, which is a measure of how quickly faults are detected by a test case, and it is calculated using the number of faults detected and the time taken to find out those faults for each test case.

$$Rf_i = \frac{\text{Number of faults detected by test case } Ti}{\text{the test case } Ti \text{ execution time}} \quad (1)$$

The value of Rf_i is the average number of faults per unit time by the test case Ti, and it is a combination of the above two factors.

For convenience of analyzing fault impact on test case prioritization, we introduce a quantification index of fault severity for each test case. Fault severity is classified five levels ranging from trivial level to critical level, and different severity values are assigned to different levels as given in Table I.

The total severity value about test case Ti is calculated

$$Sf_i = \sum_{l=1}^{r} S(l) \quad (2)$$

Where $S(l)$ represents the severity value of fault l, and r is the number of faults revealed by the test case Ti. The index of fault severity for test case Ti is defined as follows:

$$IS_i = \frac{Sf_i}{\max(Sf)} \quad (3)$$

Where $\max(Sf)$ is the highest severity value of test case among all the test cases.

TABLE I. SEVERITY VALUE OF FAULT IMPACT

Level of severity	Severity Value
trivial	2
minor	4
normal	6
major	8
critical	10

III. ANT COLONY OPTIMIZATION FOR TEST CASE PRIORITIZATION

A. *The Principle of Ant Colony Optimization*

Ant Colony Optimization is one of the adaptive mete-heuristic optimization methods, which was proposed by M. Dorigo in 1992 [22], and it was inspired by the behavior of ants in nature. During food searching, ants deposit an amount of pheromone on the traveled paths, and the pheromone dissipates over time. When other ants come to a fork in the pheromone road, they tend to follow the path with more pheromone, and the pheromone is reinforced as other ants choose the same path. Moreover, ants used short paths to food source return to nest sooner, therefore, the shorter the path, the less pheromone evaporation, the more residual pheromones. As more ants tend to follow the shorter path according to the amount of residual pheromone, ultimately all ants converge to the shortest path.

ACO method has been successfully applied to various combinatorial optimization problems such as travelling salesman problem [23], vehicle routing [24], target assignment [25], multi-objective resource allocation [26], test data generation [27] etc. Test case prioritization problem is a typical combinatorial optimization problem, our proposed idea is to optimize test case ordering based on ACO.

B. *Algorithm Design*

When using ACO to solve test case prioritization problem, the first step is to establish an ant system model about the problem, and the ant system can find solutions from this construction model, then two key rules in ACO are determined, which are pheromone updating rule and test case node selection rule, respectively. For regression testing, suppose there are n test cases in the original test suite T, which finds m faults totally, and there are also n artificial ants. In Fig. 1, Ti indicates the i th test case in T, and Lj indicates alternative test case list after an ant has selected (j-1) test cases. In the initial moments, n artificial ants are placed in the alternative test case list L1, then each of the ants randomly selects test case nodes in next lists until every ant finds all faults, and the behavior of an ant is irrelevant to all other ants. It is important to note that there is a necessary updating operation for next test case lists,

and every ant has its own lists. When the iteration is completed, an ant constructs a test case order, then the second iteration is processed according to pheromone updating rule and test case node selection rule. It probably requires several rounds of iteration until the prioritization goal is satisfied.

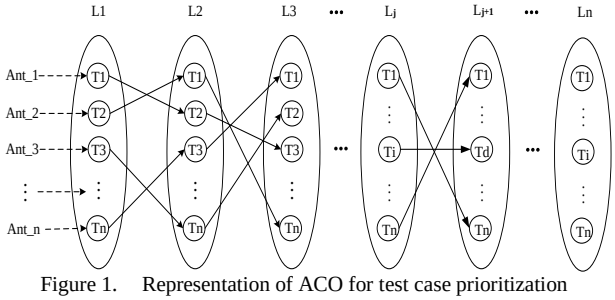

Figure 1. Representation of ACO for test case prioritization

1) Pheromone Updating Rule

When one iteration is done, the quantity of pheromone on each path is updated by the pheromone updating rule, which is given as followings:

$$\tau_{id}(t+1) = (1-\rho)\cdot\tau_{id}(t) + \Delta\tau_{id} \quad (4)$$

$$\Delta\tau_{id} = \sum_{k=1}^{n}\Delta\tau_{id}^{k} \quad (5)$$

$$\Delta\tau_{id}^{k} = \begin{cases} Q\cdot h(k) & \text{if the ant } k \text{ selects the path between } Ti \text{ and } Td \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$h(k) = \frac{\sum_{i=1}^{M_k} Rf_i^k(t)}{M_k} \quad (7)$$

Where $\rho \in (0,1)$ is the evaporation rate of pheromone; $\tau_{id}(t+1)$ is the updated quantity of pheromone between the test case node Ti and the next node Td in the (t+1)th iteration; $\tau_{id}(t)$ is the last quantity of pheromone between two nodes; $\Delta\tau_{id}$ is the total increment of pheromone on this path; n is the size of ant population; $\Delta\tau_{id}^{k}$ is the quantity of pheromone which the ant k left on this path; Q is a constant representing all pheromone that an ant possesses; $h(k)$ is the function that is related to the rate of fault detection about the ant k in the t th iteration; M_k is the number of all selected test cases when the ant k finds all faults. Two of the factors mentioned in section (2.2) are considered in the function $h(k)$, and the test case order of the ant k is better if the value of $h(k)$ is higher.

2) Test Case Node Selection Rule

Test case node selection rule determines the probability of next test case selected. Without loss of generality, assume that the ant k currently selects the *i*th test case Ti in the *j*th selection list Lj, then the *d*th test case Td will be selected in the (j+1)th selection list Lj+1 according to the following formulation (8). In the *t*th iteration, $p_{id}^{k}(t)$ is defined as the transition possibility of the ant k which finds a path from the test case node Ti to the next test case node Td. The probability depends on two factors: pheromone and visibility, which can be adjusted by the parameters α and β below. The probability of the selected test case node is higher if the pheromone in this path is more intense.

$$p_{id}^{k}(t) = \begin{cases} \dfrac{[\tau_{id}(t)]^{\alpha}\cdot[\eta_{id}(t)]^{\beta}}{\sum_{s\in L_{(j+1)}^{k}}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}(t)]^{\beta}} & if\ Td \in L_{(j+1)}^{k} \\ 0 & otherwise \end{cases} \quad (8)$$

$$\eta_{id}(t) = IS_d(t) \quad (9)$$

Where:

$\tau_{id}(t)$: pheromone intensity of the path between the node Ti and the node Td.

$\eta_{id}(t)$: visibility function which implies the heuristic information between the node Ti and the node Td.

α: parameter to regulate the influence of $\tau_{id}(t)$.

β: parameter to regulate the influence of $\eta_{id}(t)$.

$L_{(j+1)}^{k}$: the alternative test case list for ant k in the (j+1)th selection.

The value of $\eta_{id}(t)$ is determined by the index of fault severity about the test case node Td.

C. Effectiveness Metric

The goal of test case prioritization varies with different requirements, and it is necessary to assess the effectiveness of various test case prioritization results using a measure criteria. In this article, we focus on the rate of faults detected, that means the test case order is better if it detects maximum possible faults earlier. For this purpose, a measure of how rapidly a prioritized test suite detects faults was previous proposed [20, 21], which represents the weighted average of the percentage of faults detected, or APFD, during the execution of the test suite. APFD metric method is presented as follows:

Given: T, it is a test suite containing n test cases; F, it is the set of all faults revealed by T, and the number of faults is m; $\mathbf{T'}$, it is an ordering of T; TF_i, it is the position number of the first test case in T' that detects fault i.

Calculate: APFD

$$\text{APFD} = 1 - \frac{TF_1 + TF_2 + \cdots + TF_i + \cdots + TF_m}{nm} + \frac{1}{2n} \quad (10)$$

APFD indicates the rate of faults detected according to the test case ordering $\mathbf{T'}$, and the higher value of APDF means faster fault detection rate. APDF can be used a computational method of the objective function *f* in the test case prioritization problem.

IV. Experiment and Results

In this section, we present the results of our proposed algorithm to validate its effectiveness. The regression test suite contains eight test cases with the original ordering {T1, T2, T3, T4, T5, T6, T7, T8}, and these test cases cover a total of ten faults. The number of faults, execution time and fault severity value for each test case can be obtained according to historical records, which are used as inputs to the algorithm.

The Table II shows the number of faults detected by each test case, the execution time to find out those faults and index of fault severity for each test case. Four test case orders are

compared, which are original order, optimal order, random order and prioritized order using our ACO algorithm listed in Table III. The performance of those orders is evaluated by APFD, and the results are shown in Fig. 2. It shows the performance of the test case ordering using our ACO algorithm is comparable to optimal ordering, and their APFD both are 82.5%, which are superior to the results of original order and random order. Fig. 3 and Fig. 4 represent the fault severity value detected and execution time in different execution order of test cases. Optimal order and ACO order require the first three test cases executed, and 16 time units to cover total faults, and the other two approaches take more test cases and more time. For total fault coverage, original order requires the first five test cases executed and 24 time units, and random order requires the first six test cases executed and 29 time units. By comparison, it shows the effectiveness of proposed ACO algorithm in test case prioritization according to APFD and minimum execution time criteria.

TABLE II. TEST CASES, FAULTS DETECTED AND ITS EXECUTION TIME

Test case / Faults	T1	T2	T3	T4	T5	T6	T7	T8
F1 (minor)		√	√					
F2 (trivial)	√					√		√
F3 (trivial)		√		√	√	√	√	
F4 (major)	√			√				
F5 (critical)			√					
F6 (major)					√		√	
F7 (critical)	√		√					
F8 (minor)			√			√	√	
F9 (normal)	√			√		√		
F10 (normal)					√			√
Number of faults	4	2	4	3	3	4	3	2
Execution time	7	4	5	4	4	5	4	2
Index of fault severity	26	6	28	16	16	14	14	8

TABLE III. ORDER OF TEST CASES FOR VARIOUS PRIORITIZATION APPROACHES

Original Order	Optimal Order	Random Order	ACO Order
T1	T3	T7	T1
T2	T1	T5	T3
T3	T5	T4	T5
T4	T4	T6	T7
T5	T6	T1	T6
T6	T7	T3	T2
T7	T8	T8	T4
T8	T2	T2	T8

V. CONCLUSION

This paper proposes a test case prioritization approach based on ACO algorithm, which can improve the rate of fault detection for regression testing. Here, number of faults detected, execution time and fault severity are used to optimize the test case selection and prioritization process. To validate the result of our test case prioritization technique, the experiment and analysis is made in comparison of ACO with other techniques by calculating APFD value, and results indicate that the proposed approach lead to a better solution in short testing execution time. In future, the constraint relation in test case order will be considered in test case prioritization.

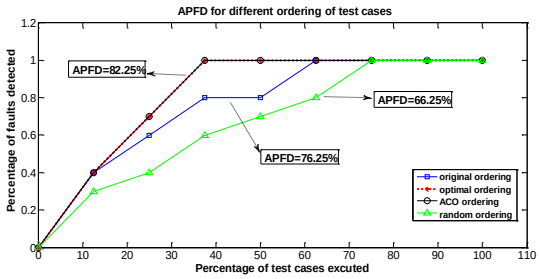

Figure 2. APFD for different ordering of test cases

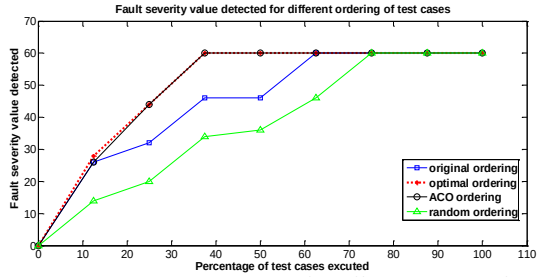

Figure 3. Fault severity value detected for different ordering of test cases

Figure 4. Execution time for different ordering of test cases

ACKNOWLEDGMENT

The authors are grateful to the referees and the Conference Committee for their valuable suggestions.

REFERENCES

[1] Yoo S, Harman M. Regression testing minimization, selection and prioritization: a survey[J]. Software Testing, Verification and Reliability, 2012,22(2):67-120.

[2] Walcott K R, Soffa M L, Kapfhammer G M, et al. Timeaware test suite prioritization[C]// Proceedings of the 2006 international symposium on Software testing and analysis. ACM, 2006:1-12.

[3] Duggal G, Suri B. Understanding regression testing techniques[C]//Proceedings of the 2nd National Conference on Challenges and Oppertunities(COIT 2008), 2008.

[4] Leung H K N, White L. Insights into regression testing[C]// Proceedings of the Conference on Software Maintenance, 1989:60-69.

[5] Rothermel G, Harrold M J. A safe, efficient regression test selection technique[J]. ACM Transactions on Software Engineering and Methodology(TOSEM), 1997,6(2):173-210.

[6] Graves T L, Harrold M J, Kim J M, et al. An empirical study of regression test selection techniques[J]. ACM Transactions on Software Engineering and Methodology(TOSEM), 2001,10(2):184-208.

[7] Harrold M J, Gupta R, Soffa M L. A methodology for controlling the size of a test suite[J]. ACM Transactions on Software Engineering and Methodology(TOSEM), 1993,2(3):270-285.

[8] Jones J A, Harrold M J. Test-suite reduction and prioritization for modified condition/decision coverage[J]. Software Engineering, IEEE Transactions on, 2003,29(3):195-209.

[9] Elbaum S, Malishevsky A G, Rothermel G. Test case prioritization: A family of empirical studies[J]. Software Engineering, IEEE Transactions on, 2002,28(2):159-182.

[10] Srivastava P R. Test case Prioritization[J]. Journal of Theoretical and Applied Information Technology, 2008,4(3):178-181.

[11] Rothermel G, Untch R H, Chu C, et al. Prioritizing test cases for regression testing[J]. Software Engineering, IEEE Transactions on, 2001, 27(10):929-948

[12] Kumar M, Sharma A, Kumar R. Optimization of test cases using soft computing techniques: a critical review[J]. WSEAS Transactions on information science and applications,2011,11(8):440-452.

[13] Krishnamoorthi R, SA Sahaaya Arul Mary. Regression test suite prioritization using genetic algorithms[J]. International Journal of Hybrid Information Technology. 2009,2(3):35-52.

[14] Kaur A, Goyal S. A genetic algorithm for regression test case prioritization using code coverage[J]. International journal on computer science and engineering, 2011,3(5):1839-1847.

[15] Kaur A, Goyal S. A Bee Colony Optimization Algorithm For Code Coverage Test Suite Prioritization[J]. International Journal of Engineering Science and Technology(IJEST),2011,3(4):2786-2795.

[16] Singh Y, Kaur A, Suri B. Test case prioritization using ant colony optimization[J]. ACM SIGSOFT Software Engineering Notes,2010,35(4):1-7.

[17] Suri B, Singhal S. Analyzing test case selection & prioritization using ACO[J]. ACM SIGSOFT Software Engineering Notes,2011,36(6):1-5.

[18] Suri B, Singhal S. Implementing Ant colony optimization for test case selection and prioritization[J]. International Journal on Computer Science and Engineering,2011,3(5):1924-1932.

[19] Zhang L, Hou S S, Guo C, et al. Time-aware test-case prioritization using integer linear programming[C]//Proceedings of the eighteenth international symposium on Software testing and analysis. ACM, 2009:213-224.

[20] Rothermel G, Untch R H, Chu C, et al. Test case prioritization: An empirical study[C]//Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on. IEEE,1999:179-188.

[21] Elbaum S, Rothermel G, Kanduri S, et al. Selecting a cost-effective test case prioritization technique[J]. Software Quality Journal, 2004,12(3):185-210.

[22] M. Dorigo. Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[23] M. Dorigo, L.M. Gambardella. Ant Colonies for the Travelling Salesman Problem[J]. BioSystems. 1997, 43(2): 73-81.

[24] K. F. Doerner, R. F. Hartl, M. Lucka. A Parallel Version of the D-Ant Algorithm for the Vehicle Routing Problem[J]. Parallel Numerics, 2005, 5: 109-118.

[25] Dongdong G, Guanghong G, Liang H, et al. Application of multi-core parallel ant colony optimization in target assignment problem[C].// Computer Application and System Modeling (ICCASM), 2010 International Conference on. IEEE, 2010, 3: V3-514-V3-518.

[26] S. K. Chaharsooghi, A. H. Meimand Kermani. An Effective Ant Colony Optimization (ACO) Algorithm for Multi-objective Resource Allocation Problem[J]. Applied Mathematics and Computation, 2008, 200(1):167-177.

[27] K. Ayari, S. Bouktif, G. Antoniol. Automatic Mutation Test Input Data Generation via Ant Colony[C]//Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM. 2007: 1074-1081.