International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 4, Issue 3, May-June 2015 **ISSN 2278-6856**

Evaluating the Effectiveness of Regression Test selection using Ant Colony Optimization

[1.]Anjali Choudhary, Tarun Dalal[2]

[1] M.Tech Scholar CBS Group of Instituion, Maharishi Dayanand University, Jhajjar, Haryana, India

[2] Asst. Professor CBS Group of Instituion, Maharishi Dayanand University, Jhajjar, Haryana, India

Abstract

Regression testing in an important phase in Software Development Life Cycle. This activity is generally performed in Software Maintenance phase. This activity is performed whenever a major or minor change is made to the application code on demand of the client or end under. These modifications can sometimes cause the application to work in inappropriate way. In order to catch bugs and errors, testers have many test cases to test the application. But in order to find out the bugs, all the test cases need to be run. Running each and every test cases is not possible every time dues to time and cost constraints. To select and prioritize the test cases, we are using algorithm, Ant Colony Optimization.

Keywords: Regression test case, selection, Prioritization, ACO, implementation.

1. INTRODUCTION

Regression testing process is carried out in maintenance phase of the software and it is one of the important phases. Regression testing is performed whenever the application code is modified. To test the modified changes whether they are working correctly or not, Regression testing is done. Sometime it is possible that due to budget and time constraint, it is not possible to re-run all the test cases in the test suite to test the modified code. In that scenario, we use test case minimization, selection and prioritization technique for Regression testing.

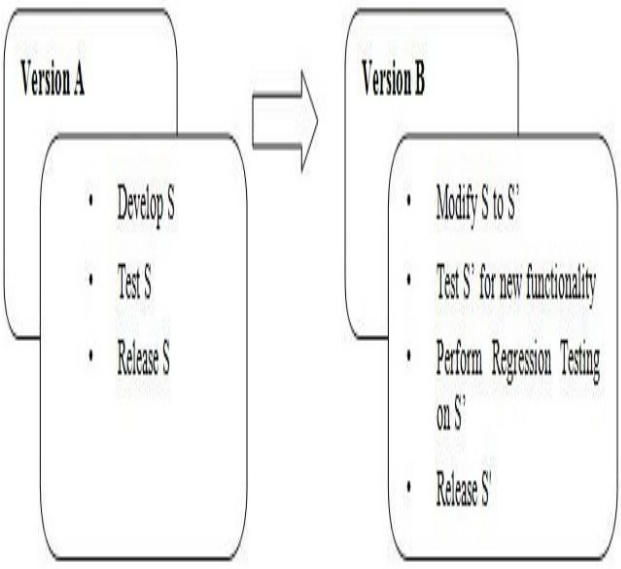


Figure 1. Regression testing process

Regression testing is performed in time constraint environment where testing is done for a limited time. In year 2010, Singh et al [7], also proposed a time constraint prioritization technique using Ant colony Optimization (ACO). The results presented in the paper provide motivation for implementing algorithm. In this paper, we will use tool Ant Colony Optimization which is generally used for solving Travelling Sales Person (TSP) problem. But we will use this tool for Regression test case selection and prioritization.

In this paper, Time and Space complexity of the algorithm is measured which provided further motivation to implement the technique. The analysis represents the effectiveness of ACO technique for test case selection and prioritization.

2. ANT COLONY OPTIMIZATION

Ant Colony Optimization technique is a set of search algorithms of Artificial Intelligence for optimal solutions. This technique was proposed by Colorni, Dorigo and Maniezzo [2][3][4]. The iconic parts of ACO are ANTS. Ants are blind and are small in size, they are still able to search for their food and reach their home using shortest path. They make use of their pheromones and antennas to be in touch with each other and find the optimal path. ACO is inspired from the behavior of ants.

ACO deals with two important processes, namely Pheromone deposition and pheromone evaporation. Pheromone deposition is the phenomena in which as the ant move from one place to another, it releases a pheromone liquid. Pheromone evaporation is the phenomena that over the time, deposited pheromone have some time period after which it evaporates and vanishes. This pheromone trail is followed by other ants by smelling which tends to follow the path with maximum pheromone trail. Point of interest is that the ant the food source and come back to the source even when they don't have eyes. Thus, ant on the shortest path is the first to come back to its nest. The process of path covering by ants is shown in Fig 1. This can be illustrated with an example demonstrated below:

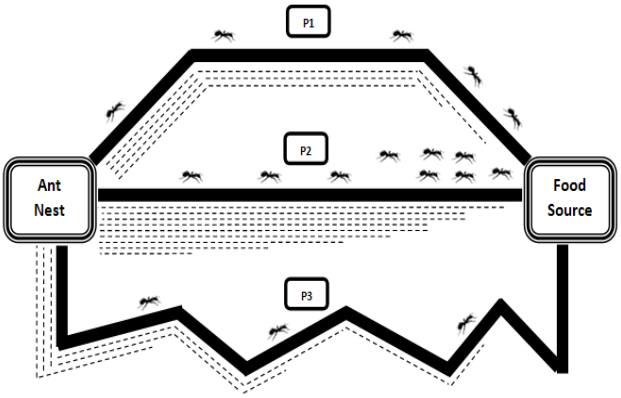


Figure 2 ACO example

Suppose there are 3 paths P1, P2 and P3, from the source(ant nest) to the destination (food source). When ant move from its nest it can choose any of the path P1, P2, or P3. After collecting food from the source ants move back to their nest. Ants on the shortest path will b the one who will reach first at their nest. Therefore more amount of pheromone will deposit on this path. New ants starting from the nest may tend to follow this shortest path and deposit more pheromone on this path. Thus, it is observed that most of the ants follow the best path and this best path has maximum pheromone deposited. This phenomenon of finding the best path is called Ant Colony Optimization.

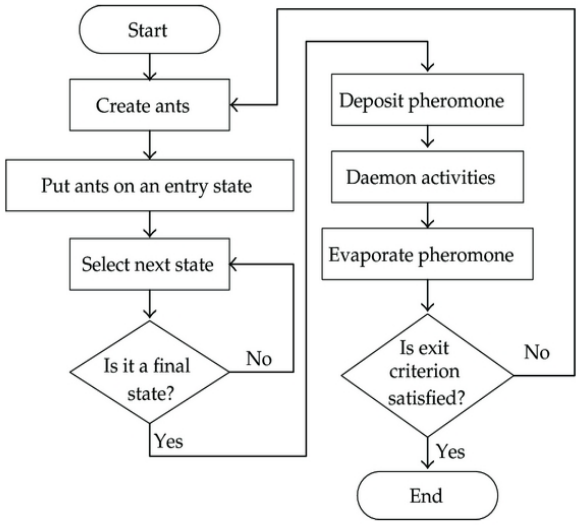


Figure 3 Flow diagram of Ant Colony Optimization representing Ants behavior

Through this behavior of ants, Dorigo proposed Ant Colony Optimization (ACO) in year 2006 [3]. ACO has been used to solve many problems like Travelling sales person problem, data mining, vehicle routing, test data generation. As proposed by Singh et al, ACO can also be used for prioritizing the test cases.

2.1Complexity of ACO algorithm

The whole effort of building an algorithm is wasted if the algorithm itself takes more time to execute all test cases in regression test suite. To prove the efficiency [1] of the algorithm in test case selection and prioritization w.r.t to time, we computed the complexity of the ACO. The Algorithm as shown in Fig.3 has a run time bounded by the time required to generate artificial ants al to an in step1, also the number of iteration for three nested loop in step2. Suppose "t" be the regression test suite with |t| no. of total test cases in it. Let "tc" be the time input by the user.

Generation of artificial ants in step1 is O(|t|) operation[5]. The inner most loop do-while calls select test case() till all faults are covered. This can repeat for maximum |t| times as there are at most |t| test cases in the test suite t. Thus, in the best case only 1 call to select the test case needs to be made to select test case(). The second nested loop i.e. for loop repeats for |t| iteration. The outermost Do while loop repeats until the execution time of the selected test case for every iteration increase beyond the user defined constraints Tc, which is constant and independent of the size of the test suite. Hence, the overall complexity of the ACO is O(TC. |t|2) for worst case, which is equivalent to O(|t|2). Select test case() itself takes constant time to execute and is independent of the size of the test suite or time constant input.

Thus, the best case complexity for the ACO algo for test selection and prioritization comes out to be O(Tc|t| equivalent to O(|t|), as the function select_test_case() is called only once to cover all faults in the application or code. This complexity is acceptable as compared to other time aware prioritization as NP-complete.

3. ACO ALGORITHM

1. Initialization:
   Set $w_i = 0$
   Set TC= MAX
   Set $Z_K = 0$ for all k=1 to n
   Generate 'n' number of artificial ants { $a_1, a_2, \ldots, a_n$ }
   $S_1, \ldots, S_n = \emptyset$
2. Do
   For k = 1 to n
   $S_K = S_K + \{ t_k \}$ //The initial test case t_k for ant a_k is the
   tmpT = t_k //starting vertex for ant a_k on the graph.
   Do
   t = Call select_test_case(a_k, tmpT)
   Update $S_k = S_k + \{t\}$
   $Z_K = Z_K +$ execution time for t
   tmpT=t
   While (total faults covered);

   EndFor
   minTime= min{ Z_K} where k=1 to n
   maxTime=max{ Z_K} where k=1 to n
   currTime= currTime + maxTime
   $Z_K = 0 \ \forall$ k = 1 to n
   Update pheromone at all the edges for the best path corresponding to minTime
   Evaporate k% (=10%) of pheromone for each edge where $w_i \geq 0$
   $S_K = \emptyset$, $\forall$ k $\in$ 1 to n
   While (TC $\geq$ currTime);

Select_test_case(a_k,tmpT)
{
If there is no pheromone deposited on the edge starting from node tmpT and ending at node $\notin S_k$ and is starting from tmpT
Then choose random edge.
Else if (number of edges starting from node tmpT and ending at node $\notin S_k$ having max pheromone is one)
Then return that edge.
Else
Return an edge randomly selected among edges starting from tmpT ending at node $\notin S_k$ having maximum pheromone.
}

Figure 4 ACO Algorithm for test case selection and prioritization proposed by Singh et al

In ACO pheromones were updated only when all the ants completed travel for a given iteration. Ant Colony system is bit different. After ants' moves from one test case to another, the pheromone on the travelled path is reduced by some value [6].

The algorithm has a parameter $q_0 \in [0,1]$. The parameter sets the boundary between two rules to select the next test case. On each step, algorithm generates a random number $0 \le q \le 1$. If $q_0 < q$, the next test case to move is selected according to (1),

$$p(c_{ij} \mid s^p) = \frac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} * \eta_{il}^{\beta}}, \forall c_{ij} \in N(s^p), \quad (1)$$

; else index J of the next test case to move is:

$$j = \arg \max_{l \in T_i^k} \{\tau_{il} * \eta_{il}^{\beta}\}, \quad (10)$$

Where

- I is the index of the current test case
- J is the index of the test case
- T_l^k is a set of test cases adjacent to the current test case.
- τ_{il} is the amount of the pheromone on the path c_{il}.
- $\eta_{il} = \frac{Q}{d_{il}}$, Q is a constant, **dil** is the distance between the test case I and l
- β is the algorithm parameter

After ant moves from one test case to another, pheromone on the path c_{il} are evaporated and updated according to

$$\tau_{ij} \leftarrow (1-\xi) * \tau_{ij} + \xi * \tau_0, \quad (11)$$

After the first ant has made the Kth step and has modified a pheromone, it does not begin the next step. It waits until all the ant completes there Kth step and modify the pheromones on their travelled path. In this process, the n+1 ant waits for the update by the ant n and so on.

3.1Evaporation and pheromone update

After all ants have completed iteration, pheromone update and global evaporation are applied to the paths traversed by the ants. In ACO, only the best to date any is allowed to update pheromone on the path. Evaporation is allowed only on path which are visited by the ants. Evaporation used equation 3,

$$\tau_{ij}^n \leftarrow (1-\rho) * \tau_{ij}^n, \quad (3)$$

And (2) and (4) are used for Global pheromone update.

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{k=m} \Delta\tau_{ij}^k, \quad (4)$$

The equations will only be applied to paths visited by the selected ants only.

3.2Pheromone Reset

If ants are struck in loop, there is an option to reset all pheromones to start values and continue run. This option might be enabled or disabled for all implemented algorithm. The selection threshold $\varsigma_{trsh}^{reset} \in [0,1]$ is a reset parameter.

If,

$$\varsigma_{trsh}^{reset} * m \ge m_{unique} \text{ and } n^{resets} \ge n_{trsh}^{reset}, \quad (12)$$

All pheromones on the path between all test cases are reset to τ_0 and search for the new shortest path.

3.2.1Implementation

Tool used in this paper is Ant Colony Algorithm. This tool is basically used for solving Travelling Sales Person problem. We have used this tool for test case prioritization and selection as well. The shortest path covered will be given the priority among many test cases; It is obvious that the test case which is covered in the shortest distance will be executed in minimum time. Testing is performed to recover the faults in the application. So it is obvious that every selected test case would recover some fault or pass the test case.

Ant Colony Algorithm tool used in this research looks as follow[6]:

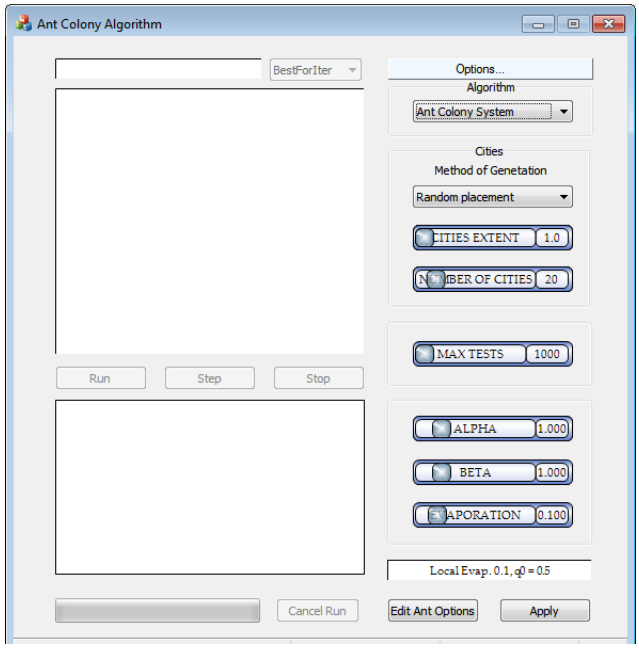


Figure 5: Ant Colony Algorithm Tool

We have provided the input as:

- Method of test case generation can be selected at Random, Automatic or Manual. We have selected the test case generation as Random.
- Test case extent is the distance between the test cases coordinates.
- Numbers of test cases are set to 10.
- Max test cases are 1000.
- Alpha i.e. Power of pheromone is set to 3.
- Beta i.e. weight on edges is set to 2.
- Evaporation is global evaporation of the pheromone 0.100.

After providing the input values we ran the application. The result is generated in the form of graph and a

diagram is created after the run as shown in fig 5.

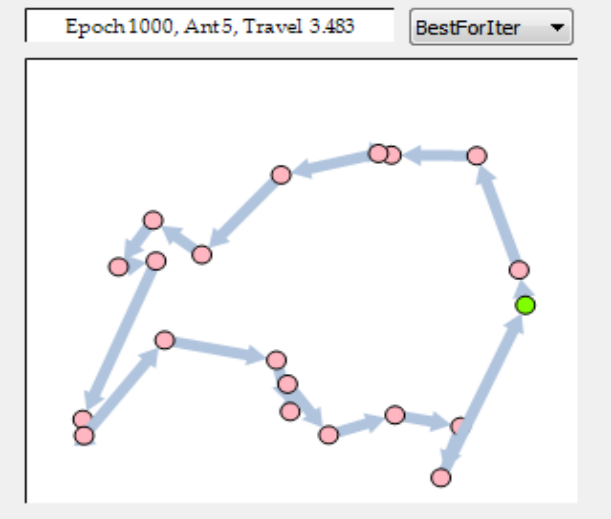


Figure 6 Final covered paths by ants

It has displayed the path which is best for iteration. Ant 5 travelled the best path to iteration. As we are representing the ants to the test cases, then test case 5 will take minimum time to execute. This test case 5 will be selected and will be prioritized. Similarly we have ranked all the test cases in the decreasing order of the distance travelled. Epoch is the particular time of period which is set to 1000 by default. The results are displayed in the table1 below:

Table 1 Prioritized order of test cases using Random Placement

Prioritized Test case	Distance travelled
5	3.483
1	3.848
8	3.87
2	3.89
0	4.45
7	4.459
9	4.555
3	4.755
6	4.792
4	5.198

We have compared the results using different test generation methods as Automatic and Manual.

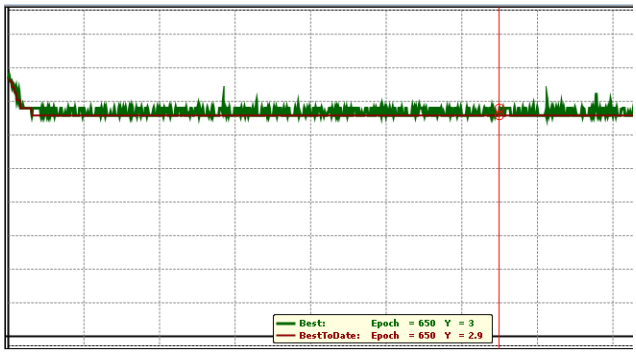
In automatic generation, ants are made to travel in circle. There are chances that two or more ants may travel the same distance. In that case it becomes difficult to prioritize the ants i.e. test case. Similar problem is faced in Manual selection of test case method generation. Therefore, preference is made for Random selection.

Table 2 Prioritized order of test cases using Automatic Placement

Prioritized Test case	Distance travelled
1,6,7	3.129
2, 8	4.01
4	4.528
9	4.668
5	5.089
3	5.36
0	7.281

After the run, test case with shortest distance travelled will be the best test case and test case which took longest distance to travel will be the worst test case. Eventually best test case took the minimum time as it traveled the shortest path and worst test case took the maximum time as it travelled the longest path. On the basis of this analysis test cases selection and prioritization is done.

A graph is also generated at the end of the run which is shown in graph1. This graph is generated from Random placement of test cases.



Graph1. Graph representing the best and best to date ant

This graph represents the best and best to date distance travelled by each artificial generated ant. It has two values currently representing Best and best to date distance of ant.

4. CONCLUSION AND FUTURE WORK

In this paper we have presented ACO technique for test selection and prioritization which leads to optimal solution for selecting test cases. Efficiency of the algorithm is also computed which has encouraged the use of ACO in time constraint test selection and prioritization. In future we wish to apply the ACO on big and more complex system problems.

References

[1] Y.Singh, A.Kaur, B.Suri, "Test case prioritization using ant colony optimization", ACM SIGSOFT Software Engineering Notes, Vol.35 No.4, pages 1-7, July 2010

[2] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies", Proceedings of ECAL'91, European Conference on Artificial Life, Elsevier Publishing, Amsterdam,

[3] M. Dorigo, "Optimization, learning and natural algorithms", Ph.D. Thesis, Politecnico diMilano, Milano, 1992.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: an autocatalytic optimizing process", Technical Report TR91-016, Politecnico di Milano (1991).
[5] T.H.Cormen, C.E.Leiserson, R.L.Rivest and C.Stein, "Introduction to Algorithms", PHI Publications, 2009 edition.
[6] www.codeproject.com
[7] Y.Singh, A.Kaur, B.Suri, "Test case prioritization using ant colony optimization", ACM SIGSOFT Software Engineering Notes, Vol.35 No.4, pages 1-7, July 2010.

AUTHOR(S)

Anjali Choudhary received the B.tech in Information technology from Rajasthan Technical University. She is now pursuing M.tech in Computer Science and Engineering from CBS group of institution from Maharishi Dayanand Univeristy, Haryana, India.

Tarun Dalal is Asst Professor in department of Computer Science and Engineering, CBS group of institution, affiliated from Maharishi Dayanand University, Haryana, India.