Test Case Suite Reduction with Classification Rule Discovery

Bhawna Jyoti[1], Aman Kumar Sharma[2]

Computer Science Department, Himachal Pradesh University, Shimla, H.P.

Abstract

In this modern age of computer infrastructure, life of a common man is software dependent. Process of identifying errors and observing performance in modified software plays a key role in maintenance of the software. Regression testing is very important as well as expensive activity to test correct functioning of the modified software. Test case suite size grows very much with the addition of test cases and there is limited testing team staff as well as testing tools to run all of test cases. Test Case Suite (TCS) minimization strategies lowers cost by reducing dimensions of a test case suite which cover all the functionalities of complete test case suite. In this study, an enhanced metaheuristic algorithm is proposed based on classification rule discovery of test cases in a test case suite. Ambha_ACO uses ant colony optimization algorithm ensembled with a decision tree to find the global optimal solution for a test case suite reduction problem. To implement our algorithm Ambha_ACO, twelve JUnit test case suites are taken and classification rules are discovered using antMiner tool. To validate results, performance parameters like Average Percentage of Fault_Detection metric,Execution time of test case suite, test case suite reduction ratio are considered. Comparative analysis of ACO with Ambha_ACO is performed and proved that Ambha_ACO gives a minimal test case suite with good APFD values as compared to ACO test case suite reduction algorithm.

Keywords –Ant colony optimization algorithm, Test case suite reduction, Fault Detection rate, Classification rules, antMiner.

NOMENCLATURE

Ant Colony Optimization (ACO), Average Percentage of Fault_Detection (APFD).

I.INTRODUCTION

The economic growth of any country mainly depends upon rapid scientific innovation and quality of the software industry. The software brings conformability in daily routines, banking, health issues, using data by innovators and in nutshell software has become part and parcel of human life. In real world scenario of maintaining software, testing team typically depend upon regression testing activity to ensure quality and reliability of modified software. For complex functionalities, regression testing techniques are strongly needed to maintain reliability and quality of modified software. [22,23,28,38]

Test case suite reduction techniques primarily focused on source-code execution coverage and fault-finding property of test case suite. Literature [12] [13] [14] revealed that various heuristic as well as metaheuristic algorithms have been implemented to solve test case suite reduction problem but there exist research gaps which can be filled by proposing an enhanced metaheuristic algorithm.

The present paper is structured as: Section II briefly tells the research gaps in the existing literature. Section III presents the basic idea about Ant Colony Optimization and proposed Ambha_ACO test case suite reduction algorithm. Section IV discloses parameters setting, metrics considered as well as experimental setup done to implement proposed algorithm. This section also reports the results and comparative study between

ACO and Ambha_ACO test case suite reduction algorithms. Finally, Section-V, provides future guidelines and concludes the paper work.

II.Research Gaps

The following table 1 is used to highlight the research gaps in existing reduction techniques:

Table 1: Research gaps in existing literature

Reduction Technique Proposed	Criteria taken	benefits	Shortcomings
H heuristic algorithm [23]	Test case requirements, essential test cases with cardinality number	Reduced test case suite obtained	Less attention on fault detection capability and local search space is exploited
GRE (greedy redundant essential) heuristic [28]	Combination of basic greedy strategy, removal of redundant(R) test cases and considering essential (E) test cases covering requirement	Reduced test case suite obtained	Only local optimal solution is found and no global solution is obtained, fault detection capability is not considered.
Hierarchal clustering technique [29]	Concept analysis is implemented by considering objects (O) and requirements(R) as attributes, context table (CT) is formulated	Smaller test case suite size is obtained	Only consider single criteria can lead to remove some important test cases from test case suite.
Enhanced tie breaking Technique [30]	Tie between test cases is removed by considering second additional criteria along with requirement coverage. Those test cases which cover few requirements will be removed to break the tie.	Fault detection rate is improved.	Time complexity is increased, small test case suite is considered.
Test case suite minimization Model [31]	Integral programming formulation is done and decision vector is found to minimize objective function	Execution time is reduced of test case suite.	Small test case suites are considered for their empirical study.
Reduction with Genetic algorithm [22]	Most Frequent test selection criteria is used along with genetic algorithm to find cardinality number	Good APFD metric is obtained	Small test case suite considered
Genetic algorithm-based Test case suite reduction [32]	Variable chromosome length with genetic operators is used.	Reduced test case is obtained	Less focus on fault detection capability.

Ant Colony optimization algorithm is widely accepted for classification rule discovery having wide range of applications like speech recognition, handwriting recognition, biometric identification, intelligent hypothermia care system, predicting degree and type of lung cancer, domain specific expert systems, mobile robot path planning and many more [18][43][44][45][46]. Classification rules based on ACO are also used to train Bayesian network classifiers. From existing literature [1][4][9], it is analysed that classification rule (CR) discovery based on natural ants can also be implemented in test case suite reduction technique. Although classical ant colony optimization-based reduction technique is implemented in the existing literature [6][7][8][10] but there are limitations of slow convergence, stagnation and poor search space exploitation. So, there is a need to propose enhanced ant colony-based algorithm on classification rule (CR) discovery in regression testing domain.

III. Proposed Algorithm

For test case suite reduction problem, we proposed an enhanced metaheuristic algorithm Ambha_ACO (using Ant Colony Optimization).

3.1. Brief Description of Ant Colony Optimization algorithms

In early 1990, Marco Dorigo coined term Ant colony optimization algorithm based on self-organising character of colony of ants [6][7][8]. These algorithms are based on construction procedures which builds solution space by using heuristic information about the given NP-hard combinatorial problem [11][19][20]. Probability(P) of ant going from source$_i$ to destination$_j$ in search of food is determined by using a transition function (TF). The transition function is described as:

$$TF_{ij} = \frac{(\acute{\emptyset}_{ij})^{\alpha}(r_{ij})^{\beta}}{\sum_{i,j=1}^{n}(\acute{\emptyset}_{ij})^{\alpha}(r_{ij})^{\beta}} \qquad (1)$$

here TF_{ij} is the function used to decide the route selected by ant based on pheromone secretion $\acute{\emptyset}_{ij}$, $(r_{ij})^{\beta}$ is the probability of selecting same path again (shortest path between source and destination) and α, β are influence parameters that ranges between [0 2]. The pheromone concentration decays(γ) varies exponentially with time(t) with initial pheromone concentration $\emptyset_0$can be given as:

$$\acute{\emptyset}_{ij}(t) = \emptyset_0 e^{-\gamma t} \qquad (2)$$

The concentration pheromone value decreases due to evaporation and is updated by using formula given below:

$$\acute{\emptyset}_{ij}{}^{t+1}_{update} = (1-\gamma)\emptyset^t_{ij} + \delta\emptyset^t_{ij} \qquad (3)$$

here $\delta\emptyset^t_{ij}$ is pheromone deposit concentration value released by ant on the travelling path. Partial current rule is constructed for *testcase$_{ij}$*using the following probability transition function:

$$P_{ij} = \frac{\vartheta_{ij}\cdot\tau_{ij}}{\sum_{i=1}^{a}\vartheta_i \cdot\sum_{j=1}^{b}\vartheta_{ij}\cdot\tau_{ij}} \qquad (4)$$

here ϑ_{ij} represents heuristic_function and τ_{ij} is amount of chemical concentration deposited on travelling path. [1,4,9]

3.2. Ambha_ACO test case suite reduction algorithm

Ambha_ACO algorithm is based on classification rule discovery which traverses set of test cases training data and finds best rule using ants foraging strategy. Test cases that are being traversed by ants removed from the list and process continues until best rule is constructed. Each ant is used to construct classification rule in the form of if (condition applied on training test case data) then (class identification). Initially ant start with empty rule and set of parameters used to construct rules. We have these criteria as a basis for test case suite reduction- **Primary criteria**: Requirement coverage by test cases, **Additional criteria's**: Test Execution environment, particular preconditions if exists to run a test case suite, Missing and duplicate test case, and non-relevant test case. The pseudocode of proposed algorithm is given below:

```
Begin
Initialize parameters of ant colony optimization;Apply test case reduction technique
Input: a setOfTestCases; TestCase_selected_list ⟶ Φ (Initial value empty)
while (uncovered test case)> set of training data (test case suite)
do
initialize pheromoneConValue ();calculate HeuristicValFun ();
BestRuleCreated ⟶ Φ (Initial value empty)
antsCreated ();RuleCreated for TestCaseSelectionList ()
Identification of class ();qualRuleFun ()
Verify quality (current Rule for selection), Best (rule in list);
learningRulesCreated ⟶ learningRule+BestRule ();
TrainingtestCaseSuite= TrainingtestCaseSuite -Test case considered for creating Rule;
while loop closed.End
Output: Reduced Test Case Suite obtained.
```

Figure 1: pseudocode of Ambha_ACO algorithm

Empty rule list (L) is created and probability of rule created by $testcase_{ij}$is described as:

$$P_{ij} = \frac{x_{ij}.\tau_{ij}}{\sum_{i=1}^{a} x_i \cdot \sum_{j=1}^{b} x_{ij}.\tau_{ij}} \quad (5)$$

Quality function (QF) is determined for rule created by test case by the following equation:

$$QF = \frac{true\ pos(TP)}{true\ pos(TP)+false\ neg(FN)} + \frac{true\ neg(TN)}{false\ pos(FP)+true\ neg(TN)} (6)$$

Concentration value of pheromone is updated by using following equation:

$$\tau(t+1) = \tau(t) * QF \quad (7)$$

3.3. Experimental setup

A java-based tool 'GUI ant miner' for extracting rules from data is used in experimental study. Ten cross-validation mechanism is used for splitting test case suite into 10 stratified partitions. Environment variables are set as follows: Ants taken = [50,500],Total no. of candidate solutions (number of iterations) that can be found by ant = [1500],
Evap_ factor = [0.90],tau (pheromone) = [1],eta (probability heuristic variable) = [1],alpha (control variable of pheromone concentration) = [1],beta (control variable of heuristic function) = [0.1], rho (concentration decay coefficient) = [0.2]. Junit test case suites used in the empirical study are taken from repositories [38][39][40][41].

Table 2: Description of Test case suites

program	branches	methods	test case classes	test cases	Version	faults
apex_hosp	665	313	74	156	2	16
st_travis	715	318	55	165	2	14
student-service	746	256	56	188	2	15
ax_bank	806	248	96	159	2	14
user_interaction	806	266	81	170	2	16
beta_lib	830	298	89	178	2	15
sin_hosp	794	232	90	220	2	14
lid_ctu	841	242	91	262	2	17
webtesting	241	282	15	350	2	12
nf_core	225	311	28	322	2	18

work_lambda_health	229	345	35	359	2	19
Alpha_lib	339	368	42	358	2	12

IV. Results

Evaluation of our proposed Ambha_ACO is done on eight performance parameters. Percentage-wise decrease or increase is also calculated for comparison between ACO and Ambha_ACO.

4.1 Test Case Suite Reduction Ratio: It is defined as size reduction of obtained test case suite as compared to original test case suite used in regression testing.

$$Test\ case\ Suite\ Reduction(TSR) = \frac{original\ Test\ case\ Suite - Reduced\ Test\ Case\ Suite}{Original\ Test\ case\ Suite}$$

Table 3: Comparison of TSR ratio of ACO and Ambha_ACO

Sr No.	Program	Test case suite reduction ratio (%) using ACO	Test case suite reduction ratio (%) using Ambha_ ACO	Increase (%)
1	apex_hosp	61.9	79	27.6
2	st_travis	55.2	70.2	27.1
3	student-service	61.3	78.8	28.4
4	ax_bank	31.6	57.3	81.3
5	user_interaction	31.3	55.1	76.0
6	beta_lib	39.3	53.08	35.0
7	sin_hosp	55.9	70.4	25.9
8	lid_ctu	59.3	73.8	24.4
9	Webtesting	61.1	76.1	24.5
10	nf_core	69.1	84.6	22.4
11	work_lambda_health	71.1	87.6	23.2
12	Alpha_lib	75.9	87.5	15.2

Subject programs nf_core (84.6), work_lambda_health (87.6) and Alpha_lib (87.5) showed very good TSR values by implementing our proposed technique as compared to traditional ACO algorithm. There is good increase in percentage of TSR i.e., 81.3 % in case of ax_bank and 76.0 % in user_interaction subject programs.

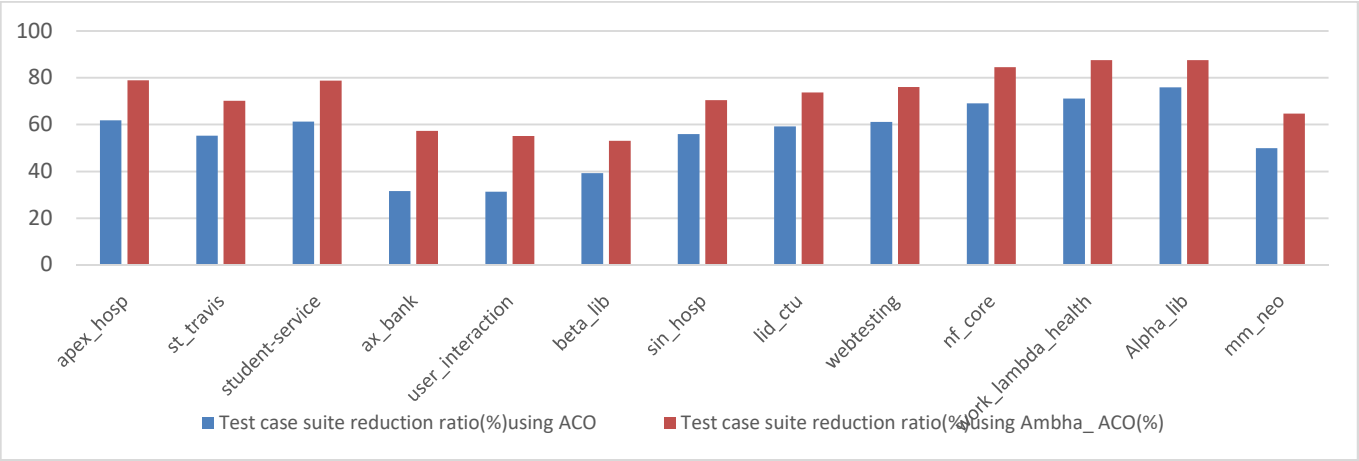

Figure 2: Comparison of Test case suite Reduction Ratio

The above Figure 2, showed that there are greater TSR ratio values obtained in each subject program as compared to traditional ACO algorithm.

4.2 Average Percentage of Fault Detection:APFD metric is calculated of a given test case suite is described as

$$APFD\ metric = 1 - \frac{1}{m * n} \sum_{i=number\ of\ test\ cases(n)}^{m\ number\ of\ faults} TF_i + \frac{1}{2n}$$

$$where\ m = number\ of\ faults, n = number\ of\ test\ cases$$

Table 4: Comparison of APFD metric of ACO and Ambha_ACO

Sr No.	program	APFD using random order of execution	APFD using ACO	APFD using Ambha_ACO	Increase in % of APFD metric (ACO/Ambha_ACO)
1	apex_hosp	0.456	0.751	0.921	22.6
2	st_travis	0.457	0.521	0.909	74.4
3	student-service	0.458	0.567	0.897	58.2
4	ax_bank	0.432	0.657	0.862	31.2
5	user_interaction	0.468	0.675	0.892	32.1
6	beta_lib	0.498	0.536	0.899	67.7
7	sin_hosp	0.412	0.599	0.91	51.9
8	lid_ctu	0.437	0.611	0.98	60.3
9	webtesting	0.456	0.654	0.899	37.4
10	nf_core	0.465	0.675	0.919	36.1
11	work_lambda_health	0.455	0.679	0.856	26.0
12	Alpha_lib	0.466	0.673	0.846	25.7

The values of APFD metric for programs apex_hosp (0.921), beta_lib (0.899) and nf_core (0.919) is good as compared to ACO and random order of execution. There is percentage increase of 74.4 (st_travis) and 67.7 (beta_lib) in reduced test case suite by Ambha_ACO as compared to ACO. Graphically, it can be presented as:

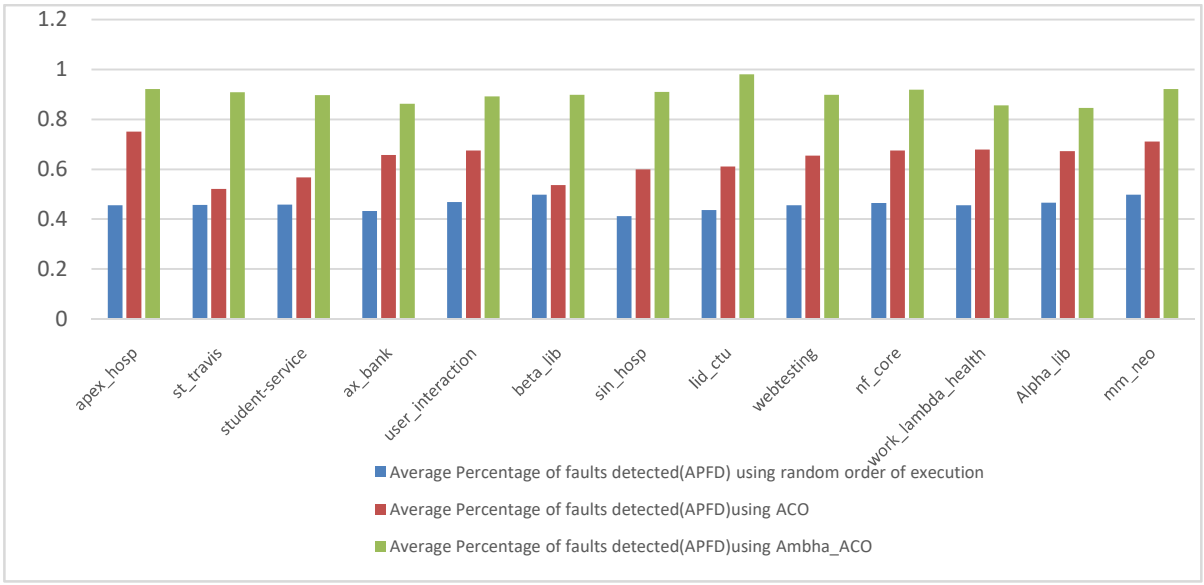

Figure 4: Comparison of APFD metric of ACO and Ambha_ACO

4.3 Execution time: It is defined as time taken by test case suite to do regression testing of subject programs. The comparative values of ACO and Ambha_ACO are described in Table 7.

Table 5: Comparison of Execution of ACO and Ambha_ACO

Sr No.	program	Execution time in sec (ACO)	Execution time in sec (Ambha_ACO)	decrease (%)
1	apex_hosp	31.1	23.9	23.1
2	st_travis	21.5	11.9	44.6
3	student-service	28.9	21.6	25.2
4	ax_bank	13.8	9.8	28.9
5	user_interaction	12.4	9.7	21.7
6	beta_lib	16.7	10.9	34.7
7	sin_hosp	18.8	12.9	31.3
8	lid_ctu	19.8	11.7	40.9
9	webtesting	21.3	12.9	39.4
10	nf_core	22.3	12.9	42.1
11	work_lambda_health	24.5	14.7	40.0
12	Alpha_lib	14.5	10.2	29.6

Execution time of various subject programs is observed and there is significant decrease in execution time in all subject programs by using our proposed approach. There is good decrease in percentage of execution time in st_travis (44.6), nf_core (42.1) and work_lambda_health (40.0) by implementing Ambha_ACO than ACO for test case suite reduction technique.

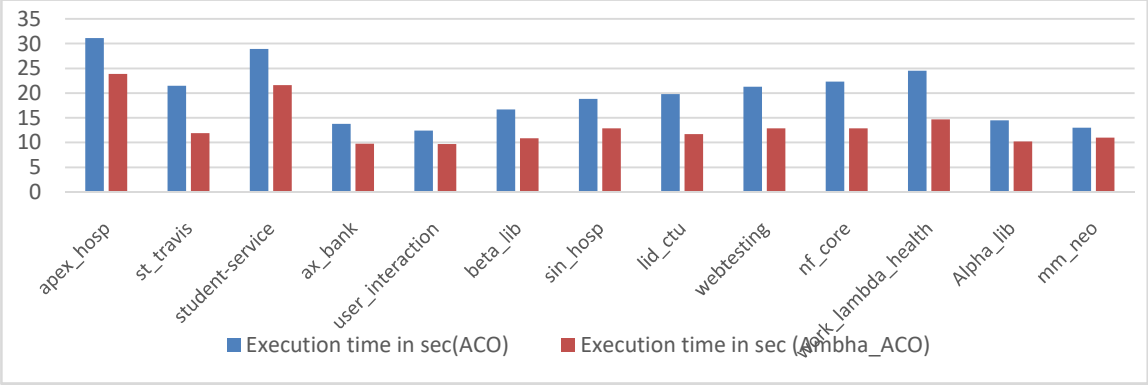

Figure 5: Comparison of execution time ACO and Ambha_ACO

4.4 : Results of NP-STATISTICAL MEASUREMENTS:

We have conducted non-parametric test for statistically prove correctness of results of proposed study. Datasets used in the present are not homogeneous, so we cannot do parametric test for comparing values of Ambha_ACO and ACO. Data is check against normality test for homogeneity of data.

1. Wilcoxon ranked signed tests: This NP-statistical test assign ranks to compare two data samples. It is calculated by using formula:

n1_pos=$\sum$n1(Ei >0) + 1/2$\sum$n2 (Ei =0);

n2_neg=$\sum$n1(Ei <0) + 1/2$\sum$n2(Ei =0); here n1_pos and n2_neg presents positive ranks and negative ranks respectively

n1_pos value is value 4.721 and n2_neg value is 7.621. Standard statistical table is used for comparison with t-values (0.242,0.412). It is proved by seeing results Ambha_ACO has significantly perform better than ACO for TSR problems in regression testing.

2. Friedman Tests: This is a non-parametric statistical test (alternate to ANOVA test) and is executed multiple times to achieve results. Find rank assigned to each block and then do statistical test as defined by following equation:

$$(av)^2 = \frac{12n}{\mathrm{m(m+1)}} \sum (r)^2 - \frac{\mathrm{m(m+1)}}{4}$$

Here (m+1) give degree of freedom and av denotes average value of ranks.

Level of significance $(\alpha) = 0.05, (av)^2 = 11.631$ exist in area of rejection. These values are compared with standard stat table and observed that Ambha_ACO performs better than ACO algorithm for TSR.

V. Conclusion

Test case suite reduction techniques are used to overcome limitations of resource as well as time constraints while doing regression testing of modified source code program. Ant Colony Optimization is a metaheuristic technique that is able to find optimal solution by exploiting search space for a given problem domain. We have used rule-based discovery classification mechanism using natural ants to obtain a reduced test case suite. An enhanced Ambha_ACO is proposed to address issues of low convergence as well as poor exploitation rate of classical ACO. Our proposed algorithm has shown very good results of test case suite reduction ratio in subject programs nf_core (84.6), work_lambda_health (87.6) and Alpha_lib (87.5) as compared to ACO. A significant decrease (%) is observed of execution time in st_travis (44.6), nf_core (42.1) and work_lambda_health (40.0) by implementing Ambha_ACO than ACO for test case suite reduction technique. There is increase in the values of APFD metric in apex_hosp (0.921), beta_lib (0.899) and nf_core (0.919) subject programs in comparison to ACO and random order of execution. Further, percentage increase of 74.4 (st_travis) and 67.7 (beta_lib) of APFD is observed in reduced test case suite by Ambha_ACO as compared to ACO. As a future work, Ambha_ACO test case suite reduction algorithm can be implemented in e-procurement based business organizations where test case suites grow exponentially with time.

References

1. 1.H. N. Al-Behadili, K. R. Ku-Mahamud, R. Sagban. “Rule pruning techniques in the ant-miner classification algorithm and its variants: A review”, in *IEEE Symposium on ISCAIE,* pp. 78-84,2018.
2. Yang XS. “Nature-Inspired Metaheuristic Algorithms”, Second Edition. Luniver Press, 2010.
3. K. Salama, A. M. Abdelbar, F. E. Otero, “Investigating evaluation measures in ant colony algorithms for learning decision tree classifiers”, in *IEEE symposium series on computational intelligence*, pp. 1146-1153, 2015.
4. S. H. Ripon, “Rule Induction and Prediction of Chronic Kidney Disease Using Boosting Classifiers, Ant-Miner and J48 Decision Tree”, *International Conference on ECCE*, pp. 1-6, IEEE, 2019.
5. K. M. Salama, F.E. Otero, “Exploring different functions for heuristics, discretization, and rule quality evaluation in ant-miner”, in *International Conference on Swarm Intelligence,* pp. 344-345, Springer, Berlin, Heidelberg, 2012.
6. M. Dorigo, V. Maniezzo, A. Colorni. “Ant system: optimization by a colony of cooperating agents,” in *IEEE Transactions on Systems, man, and cybernetics*, Part B: Cybernetics, Vol 26, No. 1, pp. 29-41, 1996.
7. A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. I. Righini, M. Trubian, “Heuristics from nature for hard combinatorial optimization problems”, in *International Transactions in Operational Research*, Vol.3 No. 1, pp.1-21, 1996.
8. T. Stiitzle, H. Hoos. “The MAX-MIN ant system and local search for the traveling salesman problem”, in *Proceedings of IEEE*

international conference on evolutionary computation, pp. 309-314, 1997.
9. F. E. Otero, A. A. Freitas, C.G. Johnson, "A new sequential covering strategy for inducing classification rules with ant colony algorithms", *IEEE Transactions on Evolutionary Computation*. Vol.17, No 1, pp. 64-76, 2012.
10. M. Dorigo, T. Stutzle. "Ant colony optimization: overview and recent advances," i*n Handbook of metaheuristics* Springer, pp. 311-351, 2019.
11. S. Katiyar, N. Ibraheem, A. Q. Ansari, "Ant colony optimization: a tutorial review," in *Proceedings of the National Conference on Advances in Power and Control*, Manav Rachna International University, Faridabad, Haryana, India, Vol. 574, Aug 2015.
12. H. Li, C. P. Lam, "Software Test Data Generation using Ant Colony Optimization," in *international conference on computational intelligence,* pp. 1-4, 2004.
13. S. Roy, S. S. Chaudhuri. "Bio-inspired ant algorithms: A review", *International Journal of Modern Education and Computer Science*, May 2013, vol. 5, issue. 4.
14. B. C. Mohan, R. Baskaran. "A survey: Ant Colony Optimization based recent research and implementation on several engineering domain", *Expert Systems with Applications*, March 2012, vol. 39, issue. 4, pp. 4618-27.
15. J. Brookhouse, F.E. Otero. "Discovering regression rules with ant colony optimization", in *Proceedings of the Companion Publication of the 2015 ACOGEC,* pp. 1005-1012. ACM, 2015.
16. R. S. Parpinelli, H. S. Lopes, A. A. Freitas, "An ant colony-based system for data mining: applications to medical data", in *Proceedings of the 3rd ACOGEC,* pp. 791-797,2001
17. K. M. Salama, A. M. Abdelbar, F. E. Otero, A. A. Freitas, "Utilizing multiple pheromones in an ant-based algorithm for continuous-attribute classification rule discovery", *Applied Soft Computing,* Jan 2013, vol. 13, issue 1, pp. 667-75.
18. M.H. Rasmy, M. El-Beltagy, M. Saleh, B. Mostafa. "A hybridized approach for feature selection using ant colony optimization and ant-miner for classification", in *8th International Conference on Informatics and Systems,* IEEE, 2012.
19. M. Dorigo, V. Maniezzo, A. Colorni., "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, man and cybernetics*, Part B: Cybernetics., Vol. 26, No. 1, pp. 129-41, Feb 1996
20. A. Helal, F. E. Otero, "Automatic design of ant-miner mixed attributes for classification rule discovery", in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 433-440. ACM, July 2017.
21. B. Liu, H. A. Abbass, B. McKay, "Density-based heuristic for rule discovery with ant-miner", in *6th Australia-Japan joint workshop on intelligent and evolutionary system,* Vol. 184, 2002.
22. C. P. Indumathi and S. Madhumathi, "Cost Aware Test case suite Reduction Algorithm for Regression Testing," in *International Conference on Trends in Electronics and Informatics*, ICEI, 2017.
23. M. J. Harrold, R. Gupta and M. L. Soffa, "A Methodology for Controlling the size of a test case suite," *ACM transactions on Software Engineering and Methodology*, Vol. 2, No. 3, July 1993.
24. http://travis.ci.com
25. http://spring.io.projects
26. http://github.com
27. http://sir.csc.ncsu.edu
28. T. Y. Chen and M. F. Lau, "A new heuristic for test case suite reduction," *Information and software Technology*, ELSEVIER, vol. 48, 1998, pp. 347-354.
29. S. Tallam and N. Gupta, "A Concept Analysis Inspired Greedy Algorithm for Test case suite Minimization," *ACM publications*, 2005.
30. J.W. Lin and C.Y. Huang, "Analysis of Test –Suite reduction with enhanced tie breaking techniques," in *Information and software technology,* ELSEVIER, 2009.
31. J. Black E. Melachrinoudis and D. Kaeli, "Bi-Criteria Models for all-uses test case suite reduction," in *Proceedings of the 26th IEEE International Conference on Software Engineering (ICSE'04)*, 2004.
32. S.K. Mohapatra and M. Pradhan, "Finding Representative Test case suite for the Test Case Reduction in Regression Testing," in *IEEE International Conference on Computer, Communication and Control*, 2015.
33. B. Suri, I. Mangal, and V. Srivastava, "Regression test case suite reduction using a hybrid technique based on BCO and genetic algorithm," in *Special Issue of International Journal of Computer Science & Informatics (IJCSI)*, pp. 2231–5292, Vol. 2, 2006.
34. I. F. Jr, X. Yang, I. Fister, and J. Brest, "A Brief Review of Nature-Inspired Algorithms for optimization", Vol. 80, No. 3, pp. 1–7, 2013.
35. C. Lin, K. Tang and G. M. Kafhammer, "Test suit reduction methods that decrease regression testing cost by identifying irreplaceable tests", Information and Software technology, Elsevier, 2014.
36. J. Bharat and V.S. Shankar Sriram, "Genetically Modified Ant Colony Optimization Based on Trust evolution in cloud computing", Indian Journal of Science and Technology, Vol 9, No 8. 2016.
37. L.D. S. Coelho and D. L. A. Bernert, "A modified ant colony optimization algorithm based on differential evolution for chaotic synchronization", Expert System with Applications, Vol 37, pp. 4198-4203, 2010.
38. P. Liu, "An efficient reduction approach to test suite", IEEE SNPD, 2014, Las Vegas, USA.
39. S.K. Mohapatra and S. Prasad, "Minimizing Test Cases to Reduce the Cost of Regression Testing", IEEE International Conference on Computing for Sustainable Global Development, 2014.
40. S. Yoo and M. Harman, "Regression Testing Minimization, Selection and Prioritization: A Survey", in Software testing, verification and reliability, Vol. 22, pp. 67-120, 2012.
41. S.K. Mohapatra and M. Pradhan, "Finding Representative Test case suite for the Test Case Reduction in Regression Testing," in *IEEE International Conference on Computer*, Communication and Control,2015.
42. S. Upadhyaya and R. Setiya, "Ant Colony Optimization: A modified Version," International Journal of Advanced Soft Computing Applications, Vol 1, No 2, 2009.
43. Ghosh, Manosij, Ritam Guha, Ram Sarkar, and Ajith Abraham, "A wrapper-filter feature selection technique based on ant colony optimization", Neural Computing and Applications, Vol. 32, No. 12 pp. 7839-7857,2020.
44. Li, H. Soleimani, and M. Zohal, M, "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives. *Journal of cleaner production*, No. 227, pp. 1161-1172, 2019.
45. D. Zhao, L. Liu, F. Yu, A. Heidari, M. Wang, Liang, and H. Chen, "Chaotic random spare ant colony optimization for multi-threshold image segmentation of 2D entropy", Knowledge-Based Systems, 216, 2021.
46. AL-Behadili, "Intelligent Hypothermia Care System using Ant Colony Optimization for Rules Prediction", Journal of University of

Babylon for Pure and Applied Sciences, Vol 26, No. 2, pp. 47-56, 2018.