A Comprehensive Comparison of Ant Colony and Hybrid Particle Swarm Optimization Algorithms Through Test Case Selection

Arun Prakash Agrawal and Arvinder Kaur

Abstract The focus of this paper is towards comparing the performance of two metaheuristic algorithms, namely Ant Colony and Hybrid Particle Swarm Optimization. The domain of enquiry in this paper is Test Case Selection, which has a great relevance in software engineering and requires a good treatment for the effective utilization of the software. Extensive experiments are performed using the standard flex object from SIR repository. Experiments are conducted using Matlab, where Execution time and Fault Coverage are considered as quality measure, is reported in this paper which is utilized for the analysis. The underlying motivation of this paper is to create awareness in two aspects: Comparing the performance of metaheuristic algorithms and demonstrating the significance of test case selection in software engineering.

Keywords Optimization · Meta-heuristics · Ant colony optimization · Particle swarm optimization · Regression testing

1 Introduction

Every software system enters into the maintenance phase after release and keeps evolving continuously to provide the functionality required and to incorporate changing customer needs. Regression testing is an activity that tries to find any bugs introduced during the maintenance phase. One approach is to re-run all the test

A.P. Agrawal (✉)
Department of Computer Science and Engineering, Amity University,
Noida, Uttar Pradesh, India
e-mail: apagrawal@amity.edu

A. Kaur
University School of Information and Communication Technology, GGSIP University,
New Delhi, India
e-mail: arvinderkaurtakkar@yahoo.com

S.C. Satapathy et al. (eds.), *Data Engineering and Intelligent Computing*,
Advances in Intelligent Systems and Computing 542,
DOI 10.1007/978-981-10-3223-3_38

cases available from an earlier version of the software system [1]. But it is often too costly in terms of time and effort required to rerun all the test cases and is practically infeasible [2]. Two approaches have therefore emerged to optimize the regression test suite—Regression Test Case Selection and Prioritization [3]. The focus of this paper is on Regression Test Case Selection for re-execution to reduce the overall effort. Selecting the test cases would give the opportunity to optimize some performance goals like minimize execution time and maximize fault coverage.

The objective of this paper is to use two nature inspired metaheuristic techniques—Ant Colony Optimization and Hybrid Particle Swarm Optimization in testing for the purpose of minimizing the time required in regression testing by selecting the test cases and compare their performance. These two algorithms were empirically evaluated on flex object from the SIR repository and results of both the algorithms are compared on the basis of Fault Coverage and Execution Time. Results indicate that hybrid Particle Swarm Optimization outperforms the Ant Colony Optimization algorithm.

Rest of the paper is organized as follows: Sect. 2 briefly discusses the Regression Testing and states the Regression Test Case selection Problem. Sections 3 and 4 give an overview of the Ant Colony Optimization and Hybrid Particle Swarm Optimization algorithms respectively. Section 5 presents experimental design and results obtained results. Section 6 discusses the results. Finally Conclusion and Future work conclude the paper.

2 Problem Definition

2.1 Regression Testing

Primary purpose of Regression Testing is to increase the performance of software systems in terms of productivity and efficiency and to ensure quality assurance of applications [4]. The main objective of regression testing is to assure that the modification in one part of the software does not adversely affect the other parts of the software [5].

2.2 Regression Test Case Selection

Regression Test Case Selection is choosing a subset of test cases from the existing set of test cases that are necessary to validate the modified software. It reduces the testing cost by reducing the number of test cases to test the modified program [6, 7].

We consider following two criteria for the selection of test cases:

2.2.1 Fault Coverage

The aim is to maximize the Fault coverage. Let T = {T_I, T_2, …, Tn} indicates a test suite and F = {F_l,F_2, …,Fm} indicates faults covered by a test case [8]. F(Ti) indicates a function which returns the subset of faults covered by a test case. Then, the fault coverage of a test suite is given by

$$\text{Fault Coverage} = 100 * \sum_{t=1}^{s} \bigcup_{t=1}^{s} \{F(Ti)\}/k \tag{1}$$

where s reflects the selected test cases and k is the total number of selected test cases.

2.2.2 Execution Time

The execution time should be minimized. Execution time is defined as the total time required to execute a test [9]. The total Execution time of selected test cases is given by

$$\sum_{t=1}^{s} Time \tag{2}$$

where s = selected test cases.

However test case selection could reduce the rate of detection of faults because the effectiveness of a test suite could decrease with the decrease in the size of a test suite.

3 Ant Colony Optimization for Optimizing Test Suites

Ants demonstrate complexity in their social behavior which has been attracting the attention of humans since a long time. Apparently the most marked behavior is the formation of ant streets. The most astonishing behavioral pattern demonstrated by ants is that they are able to find the shortest path from its nest to food source. It is proven that this behavior is the result of communication by a hormone called pheromone, an odorous chemical substance ants deposit and smell [10]. Computer scientists are simulating real ants in artificial ants and one such example is ant colony optimization.

Description of Algorithm

1. Test Cases are treated as a graph.
2. Each node on the graph has a specific score.
3. All ants start from the same place i.e. first node.
4. Ants deposit pheromone on the paths i.e. edges between two nodes.
5. Ants move on any node within 20 nodes from node number.
6. Ant follows route with $[\tau]*[\eta]$ probability.
7. η = (Number of faults)/(execution_time).
8. $\tau = \tau + \tau_0 * \rho$.
9. where $\rho = 0.9$.
10. $\tau_0 = 1/Cnn$; where Cnn is distance between current and next node.
11. Ants can only move forward.
12. Pheromone is appended on the paths.

4 Hybrid Particle Swarm Optimization for Test Case Selection

Particle Swarm Optimization is a population based memory using optimization technique. In PSO, each particle has its own position vector and velocity vector [11]. Position vector represents a possible solution to a problem and also a velocity vector. In testing, position means rank assigned to a test case in a test suite, whereas the velocity is the coverage or the execution time taken by a test case. Every particle stores its best position seen so far and also the best global position obtained through interaction with its neighbor particles. Particle Swarm Optimization algorithm guides its search by adjusting the velocity vector and Position of particles. Objective function is responsible for moving the particles. Particles which are very far from the optimal solution have higher velocity as compared to the particles that are very near to the optimal solution. Many variants of PSO with Genetic Algorithm, Gravitational Search Algorithm, Krill Herd and Cuckoo Search Algorithm etc. has been used by previous researchers to solve many optimization problems like frame selection in image processing, aircraft landing, feature selection in software defect prediction, quadratic assignment problem etc. [11–19].

This paper presents Hybrid PSO with Genetic Algorithm for test case selection. The algorithm uses Crossover and Mutation operator of genetic Algorithm and uses the Velocity Vector of PSO to guide the particles to Global Best Solution.

Description of Algorithm

1. Read Test cases from the text file and choose them randomly from the test suite.
2. Read execution time and statements covered by each test case.
3. Set MAX_VEL, MIN_VEL, MAX_LEN, MIN_LEN, TOTAL_PARTICLES and MAX_ITER.

4. Initialize each particle with zero Velocity and random Length. Each Particle contains set of Test Cases.
5. for i = 0 to MAX_ITER.
    i. Evaluate Fitness of Each Particle.
    ii. Use Bubble sort to sort Particles in the ascending order of their fitness level.
    iii. Set Velocity of Each Particle.
    iv. Update Each Particle on the basis of the changes required (Particle with bad fitness level have higher changes).
    v. Apply Mutation with 1, 2 and 5% rate on each particle.
    vi. Determine Global and Local Optimal Values.

5 Experimental Design and Results

Our Empirical study addresses the following research questions:

Table 1 Results of ant colony optimization

Selected test cases	Execution time of selected test cases	Execution time for program	Fault coverage
T1, T3, T12, T17, T19, T21, T37, T54, T69, T77, T78, T85, T89, T106, T125, T129, T146, T153, T168, T178, T192, T199, T217, T236, T237, T243, T247, T266, TT267, T271, T289, T304, T314, T325, T343, T358, T368, T375, T379, T398, T416, T433, T438, T441, T459, T467, T477, T487, T488, T495, T502, T519, T523, T539, T554, T559, T563, T564, T566, T567	14.48	0.35	16

Table 2 Result of Hybrid PSO at mutation probability 1, 2 and 5%

Selected test cases	Execution time of selected test cases	Execution time for program	Fault coverage
Mutation probability 1%			
T303, T258, T164	0.872175	30.81	16
Mutation probability 2%			
T553, T308, T258, T414	1.07353	33.455	16
Mutation probability 5%			
T12, T412, T118, T308	1.06982	33.66	16

Table 3 Combined Results of ACO and Hybrid PSO

Algorithm	Mutation probability	No. of test cases selected	Total execution time of selected test cases	Total no. of faults covered	Total no of test cases	Total execution time of all test cases	Total no. of faults	%age of test cases selected	%age of faults covered	Execution time of Algo.	%age of time required
ACO	N/A	60	14.48	16	567	170.4	19	10.58	84.21	0.35	8.7
Hybrid PSO	1%	3	0.87	16	567	170.4	19	0.52	84.21	30.81	18.5
Hybrid PSO	2%	4	1.07	16	567	170.4	19	0.70	84.21	33.45	20.2
Hybrid PSO	5%	4	1.06	16	567	170.4	19	0.70	84.21	33.66	20.3

- *RQ1: Which algorithm is optimal for Regression Test Case selection?*
- *RQ2: How much saving can we achieve by Regression Test Case selection for the benchmark problem?*

We have used Flex—Fast Lexical Analyzer as the object under test which is available in open source from Software Artifact Infrastructure Repository [20]. It consists of 567 test cases and 19 seeded faults and execution time of each test case. Test cases have been numbered from T1 to T567 and faults have been numbered from f1 to f19. We ran both the algorithm on the input data set and collected results. We have considered bi-objective optimization as we want to maximize the fault coverage and minimize the execution time.

Table 1 above displays the results obtained upon execution of Ant Colony Optimization Algorithm on the input data set.

Table 2 above display the best results obtained upon execution of Hybrid Particle Swarm Optimization algorithm for particle size 30 and mutation probabilities of 1%, 2% and 5% respectively. Particle size was kept constant at 30 in each run and algorithm was executed 30 times for 500 numbers of iterations to avoid any biases.

6 Discussion on Results

Table 3 above displays the combined results of both the algorithms. It can easily be seen from the statistics below that the results are quite satisfactory. In case of ACO less than 11% of test cases have been able to detect 84.2% of the faults and requires only 8.7% of total time including the running time of the algorithm. Hybrid PSO algorithm has been run for 30 particles and for 500 iterations in each run and the best of 30 runs has been taken as the observed result as follows for Mutation Probability 1%, 2% and 5% respectively. Hybrid PSO is able to detect 84.21% faults with only 0.7% of the total test cases which is a very cost efficient in comparison of ACO, though the running time of Hybrid PSO is much more than ACO. Although hundred percent of the faults have not been identified this is still a great achievement since the tradeoff is quite high.

It is evident from the results that both the algorithms are optimal for regression test case selection which answers our research question 1. Hybrid PSO outperforms ACO in number of test cases selected but requires more time to run itself than ACO. We can save around 90% of the execution time through test case selection which answers our research question 2.

7 Conclusion and Future Scope

Primary focus of this paper was to present a comprehensive comparison of two meta-heuristic algorithms in the domain of software testing. Extensive experiments were conducted on the benchmark object flex and results obtained answered the

research questions in study. This allows future researchers to conduct this kind of study for number of metaheuristic algorithms on number of benchmark problems.

References

1. Mirarab, S., Akhlaghi, S., Tahvildari, L.: Size-constrained regression test case selection using multi-criteria optimization. IEEE Trans. Softw. Eng. **38**(4), 936–956 (2012)
2. Rothermel, G., Harrold, M.J., Dedhia, J.: Regression test selection for C++ software. Softw. Test. Verif. Reliab. **10**(2), 77–109 (2000)
3. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: a survey. Softw. Test. Verif. Reliab. **22**(2), 67–120 (2012)
4. Mao, C.: Built-in regression testing for component-based software systems. In: 31st Annual International on Computer Software and Applications Conference, 2007. COMPSAC 2007, vol. 2, pp. 723–728. IEEE (2007)
5. Ali, A., Nadeem, A., Iqbal, M.Z.Z., Usman, M.: Regression testing based on UML design models. In: 13th Pacific Rim International Symposium on Dependable Computing, 2007. PRDC 2007, pp. 85–88. IEEE (2007)
6. Nagar, R., Kumar, A., Singh, G.P., Kumar, S.: Test case selection and prioritization using cuckoos search algorithm. In: International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), pp. 283–288, IEEE (2015)
7. Jeffrey, D., Gupta, N.: Experiments with test case prioritization using relevant slices. J. Syst. Softw. **81**(2), 196–221 (2008)
8. Kaur, A., Goyal, S.: A bee colony optimization algorithm for fault coverage based regression test suite prioritization. Int. J. Adv. Sci. Technol. **29**, 17–30 (2011)
9. Kumar, M., Sharma, A., Kumar, R.: An empirical evaluation of a three-tier conduit framework for multifaceted test case classification and selection using fuzzy-ant colony optimisation approach. Softw. Pract. Exp. **45**(7), 949–971 (2015)
10. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
11. Liu, B., Wang, L., Jin, Y.H.: An effective hybrid pso-based algorithm for flow shop scheduling with limited buffers. Comput. Oper. Res. **35**(9), 2791–2806 (2008)
12. Apostolopoulos, T., Vlachos, A.: Application of the firefly algorithm for solving the economic emissions load dispatch problem. Int. J. Comb. (2011)
13. Chang, X., Yi, P., Zhang, Q.: Key frames extraction from human motion capture data based on hybrid particle swarm optimization algorithm. In: Recent Developments in Intelligent Information and Database Systems, pp. 335–342. Springer International Publishing (2016)
14. Wang, G.G., Gandomi, A.H., Alavi, A.H., Deb, S.: A hybrid method based on krill herd and quantum-behaved particle swarm optimization. Neural Comput. Appl. **27**(4), 989–1006 (2016)
15. Girish, B.S.: An efficient hybrid particle swarm optimization algorithm in a rolling horizon framework for the aircraft landing problem. Appl. Soft. Comput. **44**, 200–221 (2016)
16. Cui, G., Qin, L., Liu, S., Wang, Y., Zhang, X., Cao, X.: Modified PSO algorithm for solving planar graph colouring problem. Prog. Nat. Sci. **18**(3), 353–357 (2008)
17. Kakkar, M., Jain, S.: Feature selection in software defect prediction: a comparative study. In: 2016 6th International Conference-Cloud System and Big Data Engineering (Confluence), pp. 658–663. IEEE (2016)
18. Tayarani, N.M.H., Yao, X., Xu, H.: Meta-heuristic algorithms in car engine design: a literature survey. IEEE Trans. Evol. Comput. **19**(5), 609–629 (2015)

19. Agrawal, A.P., Kaur, A.: A comparative analysis of memory using and memory less algorithms for quadratic assignment problem. In: 2014 5th International Conference on Confluence the Next Generation Information Technology Summit (Confluence), pp. 815–820. IEEE (2014)
20. Do, H., Elbaum, S., Rothermel, G.: Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact. Empir. Softw. Eng. **10**(4), 405–435 (2005)