A Framework for Optimization of Software Test Cases Generation using Cuckoo Search Algorithm

Sanjiv Sharma
Department of Computer Science & Engineering
KIET Group of Institutions,
Ghaziabad, UP, India
Email: sanjiv.sharma@kiet.edu

S. A. M. Rizvi
Department of Computer Science
Jamia Millia Islamia,Jamia Nagar
New Delhi, India
Email:samsam_rizvi@yahoo.com

Vineet Sharma
Department of Computer Science & Engineering
KIET Group of Institutions,
Ghaziabad, UP, India
Email: vineet.sharma@kiet.edu

***Abstract*- Software testing is the most important phase of the software development lifecycle in the software industry. It is done to make sure that developed software is defect free; behavior of the software is same as expected and includes the generation of test data that satisfies some adequacy criteria like statement coverage, branch coverage, path coverage etc. This task is costly and time-consuming so there is an urge for automation of this process. In recent years various meta-heuristic based nature-inspired algorithms are applied in various fields of software engineering. This article proposes a framework for the generation of an optimal set of test cases using a meta-heuristic based optimization algorithm called Cuckoo Search Algorithm as well as an overall algorithm for the same.**

Keywords- Software Testing, Cuckoo Search Algorithm, Le´vy Flight

I. INTRODUCTION

Software testing is the most important phase of the software development lifecycle in the software industry. It is the process through which we ensure software is reliable and trustworthy. Software testing consumes approximately 50 percent of the total efforts needed for development of software by a software development organization. These efforts include cost, time and manpower. In software testing, one generates test data/test cases, which further works as input to the concerned software, now behavior of the software on those inputs are examined to check whether it satisfies the requirement stipulated in the document called Software Requirement Specification (SRS) or not. Software testing can be done manual or automatic. Through automatic software testing, we can minimize manual labor and improve the overall testing efficiency by reducing the testing execution time as well as increasing the fault exposure and coverage ratio. Testing may be of two types, structural testing, and functional testing. Structural testing is also known as white box testing. In this testing generation of test cases is based upon the internal structure of the program. Structural testing can be performed at the unit level, integration level or system level. There are various testing adequacy criteria available in this category like statement coverage, branch coverage, condition coverage, path coverage etc. Functional testing is also called black box testing. In this type of testing, test cases are generated using the functionality of the software under test. In this internal structure of the software is not concerned for the testing purpose. Functional testing is used to test the functionality of the system against functional requirement or specifications. In this type of testing some input is given to the program and then the output is examined without considering the internal processing of the program. It is generally performed at the system level testing or acceptance testing. Structural testing is cost-effective than functional testing.

In recent years Search-Based Software Testing (SBST) has received great attention from the researchers and professionals of software testing community. In SBST Meta-Heuristic based Search Algorithms (MHSA) are used to generate test data. Empirical analysis has proven that SBST can outperform the traditional testing methods [1]. MHSA is a collective name of algorithms for Hill Climbing Algorithm (HCA), Simulated Annealing (SA), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Cuckoo Search Algorithm (CSA), Artificial Bee Algorithm (ABA), Particle Swarm Optimization (PSO), Firefly Algorithm (FA), Bat Algorithm (BA), Harmony Search (HS) and many more [2].

CSA is an optimization algorithm which is based upon the natural behavior of some cuckoo species having brood parasitic nature and host bird. It also has some flavor of fruit flies using the concept of Le´vy flight. Yang et al. [3] developed this algorithm in 2009. CSA provides more cost-effective and reliable solutions than any other MHSA since it provides a fine balance between randomness and convergence with less number of control parameters [3]. This paper proposes a framework for the generation of an optimal set of test cases using a meta-heuristic based Cuckoo Search Algorithm as well as an overall algorithm for the same.

This research paper is organized in five sections. Introduction of the paper is discussed in section1. Section 2 discusses the basics of Cuckoo Search Algorithm. In section 3 optimizations done using CSA in different engineering fields are discussed. Section 4 discusses the proposed framework and the last section of the paper discusses conclusion and future work.

II. CUCKOO SEARCH ALGORITHM

Cuckoo Search algorithm (CSA) is a nature-inspired population-based stochastic algorithm and it is used for optimization [3]. In this search, a parasitic cuckoo lay her eggs in the nest of other host birds. To achieve this cuckoo do the phenotype matching between cuckoo and host eggs by mimicking the visual pattern of the host eggs. Cuckoo generally chooses such type of nest in which host bird has laid their egg recently. In general cuckoo's eggs, hatch in less time compared to host bird's eggs. When the first chick of the cuckoo comes outside the egg, it blindly propels other eggs outside the nest by following its natural instinct. This activity of the cuckoo chick increases its stake of the food supplied by the host bird. The algorithm proposed in [3], mimics the behavior of brood parasitism for some birds of cuckoo species and assumes that cuckoo puts one egg in a nest each time. The nest chosen by cuckoo to lay egg is random. Nest with the best quality of eggs carryovers to the next generation. There is a probability $P_a \in [0,1]$ of eggs being identified by the host bird. There is a number of nests in the cuckoo search. In a nest, an egg represents a solution and a cuckoo egg represents a new solution. If this new solution is better than the previous solution, then it replaces the worst solution among the available solution. In extended cuckoo search, a single nest may contain more than one egg which is equivalent to a set of solutions.

Cuckoo Search Algorithm is a meta-heuristic and population-based algorithm. It is used to solve the optimization problem. This algorithm based upon following three simple rules.

1. Every cuckoo randomly selects a nest to lay eggs.

2. From a fixed number of available host nests, the nest with the best quality of eggs will carryover to the next generation.

3. There is a probability pa ϵ [0,1] of each egg being identified by the host bird from a fixed number of available eggs. In this situation host bird has two choices. It may abandon the nest and make a new one or get rid of the egg.

Performance of CSA can be refined by using Le´vy flight in place of plain random walk [3]. Based upon the above three rules, Yang et al. developed Cuckoo Search Algorithm (Algorithm 1).

Algorithm 1:

1. Set the values of the objective function, initial population, step size, range of inputs and maximum generation
2. Initialize the population using initial population
3. Repeat steps from 4 to 11 until the number of iteration exceeds the maximum generation or stopping criterion reached
4. Select a cuckoo randomly (say i) and generate a new solution using Le´vy Flight
5. Calculate fitness value (F_i) of the solution using objective function
6. Randomly select a nest from available nest(say j)
7. **if** fitness value (F_i) is better than fitness value (F_j) **then**
8. Replace j with the new solution
9. **end if**
10. Abandon a fraction (P_a) of the worst solution
11. Keep track of the best solutions, rank them and find the current best
12. List out the final result

Le´vy flight

In nature, it has been observed that animal/bird search for food is in forage path. It is a random walk because of next move dependents upon the current state and transition probability to the next location as well. These things can be modeled mathematically using Le´vy flight.

$$x_i^{(t+1)} = x_i(t) + \alpha \oplus Le´vy(\lambda) \quad (1)$$

Where α denotes the step size and its values is always positive. It is related to scale of the problem in hand. Generally value of α is 1. The symbol $\oplus$ denotes the entry-wise multiplications. Le´vy-flight used for random walk and its random steps can be find using following Le´vy distribution formula.

$$Le´vy \sim u = t^{-\lambda}, (1 < \lambda \leq 3) \quad (2)$$

III. RELATED WORK

As already discussed in the above section optimization of the test case can improve the performance of the software testing and reduce the cost of software development. This section discusses the recent work done in the area of optimization problems using CSA in different fields of engineering.

In 2010 Yang et al [4] used CSA for optimal solution of complex engineering design problems like structure of the welded beam structure and design of the springs. In this study authors have found that result generated by CSA is better than Particle Swarm Optimization algorithm. In 2011 Valian et al [5] proposed an approach for fine-tuning of the CSA and after that enhancement, it is used in training of feedforward neural network for classification of Iris and Breast cancer classification problems. In 2012 Chandrasekaran et al [6] solved the famous problem of unit commitment in power system using a modified CSA algorithm. This modified CSA was integrated with fuzzy system. In 2013 Gandomi et al [7] used CSA to solve truss structure optimization problem and compared results with state of art optimization algorithms and found that result produced by CSA are better than others. In 2012 Srivastava et al [8] used CSA along with Tabu search in the generation of automated test

data. In this article, the result shows that proposed approach is more effective in the generation of optimal test cases than various earlier proposed approaches. In 2014 Chiranjib et al [9] found out optimized vehicle route from a given graph based routes and also solved famous Traveling Salesman problem using discrete version of CSA. Solution obtained from above approach were also compared with solution obtained from other two approaches called Intelligent Water Drop and Ant Colony Optimization. In 2014 Jeya et al [10] used the memetic algorithm with CSA to optimize the number of test cases with the mutant score and branch coverage based specified test adequacy criteria. In 2016 Liang et al [11] used CSA in the prediction of the surface roughness of abrasive water jet and find that CSA outperformed two famous approaches called Artificial Neural Network (ANN) and Support Vector Machine (SVM). In 2015 Naseer et al [12] have implemented a CSA based pairwise strategy for Combinatorial Testing. In 2016 Khari et al. [13] used CSA for generating test data and compare the result with the Hill Climbing algorithm. In 2017 Kumar et al. [14] used CSA and Particle Swarm Optimization (PSO). In this article, the author used PSO for optimization of test cases and prioritized those by using CSA. In 2016 Neenu et al. [15] Used Cuckoo Search Algorithm to find out optimal scheduling of different tasks in the cloud by allocating minimum time slots for their execution. In 2018 Dharmendra [16] used a blend of cuckoo search and ant colony optimization algorithms to find out optimal schedule, so that available resources can be properly utilized. In 2018 Xin and Shenghua [17] used cuckoo search algorithm for optimal train control strategy so that time and energy can be saved. In 2012 Anna and Simon [18] have used CSA in the optimization of simulation of an engine manufacturing line. In this article performance of CSA is also compared with a benchmark algorithm for pareto optimization algorithm NSGA-II. In 2013 Xin and Suash [19] developed modified cuckoo search algorithm for multiobjective optimization problem and used it in structural design (beam design) and disc brake design problem. In 2014 Bhandari et al. [20] used a blend of cuckoo search algorithm and wind driven optimization in image segmentation, so that meaningful information can be extract optimally. In 2013 A.R. Yildiz [21] used cuckoo search algorithm in milling optimization problem in manufacturing field and compare its solution with other optimization techniques like ant colony optimization algorithm , immune algorithm, particle swarm optimization and genetic algorithm. In 2012 Mahajan et al. [22] have applied the genetic algorithm in the generation of efficient test cases for data flow testing. In 2016 Sapna and Monika [23] have applied the genetic algorithm, which used the concept of dominance and branch distance to find the optimal suite of test cases for data flow testing and outperformed random search technique for the same. In 2017 Sumit et al. [24] developed a hybrid optimization algorithm called adaptive PSO-GA. This algorithm is a combination of particle swarm optimization and genetic algorithm. In this paper, this hybrid algorithm has been used to find out optimal set of test cases for data flow testing. In a comparative study, it claims that this innovative algorithm outperformed genetic algorithm, particle swarm optimization, ant colony optimization, and differential evolution. In 2017 et al. [25] have used an algorithm called modified particle swarm optimization called accelerating particle swarm optimization (APSO) algorithm for generation of optimized test cases for data flow testing. This modified algorithm outperformed random search and genetic search algorithm for several benchmarked programs.

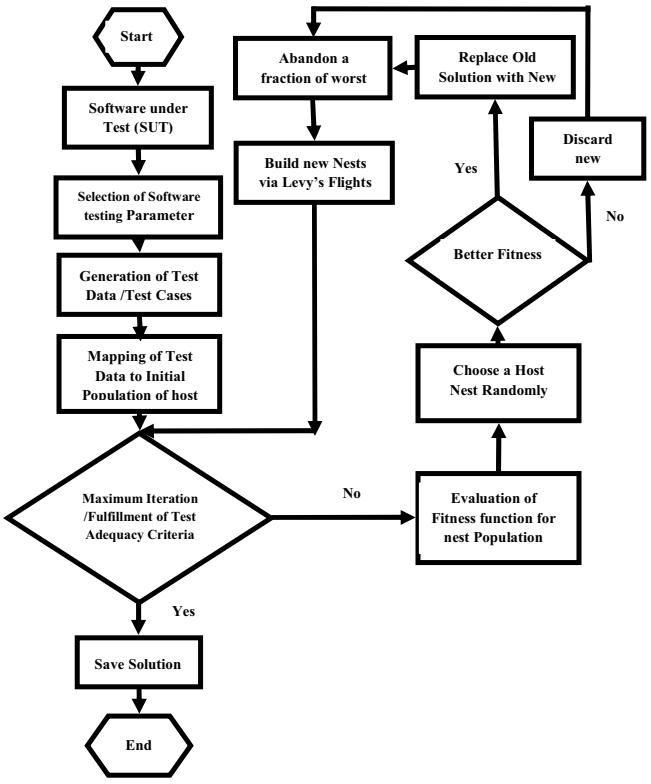

Fig. 1. Flowchart of Proposed Framework

IV. PROPOSED FRAMEWORK

In this paper, we have proposed a framework which may be used to generate an optimal set of test cases for white box testing (dynamic software testing) using cuckoo search algorithm. Flowchart of this proposed framework is given in Fig. 1 and corresponding algorithm is discussed in algorithm 2. The software under test can be transferred in suitable representation like control flow graph, control dependency graph, def-use pairs etc which can be further mapped with input requirement of CSA. and work as an input for initial population of the solution. Further, we need an objective function to evaluate the intermediate solution to determine the quality of it. If the new solution is better than the previous one, then it is used in

generation of new population, otherwise, previous solution is considered for creation of new population. CSA works in the repeated manner in order to refine solution until stipulated adequacy criteria for software testing meet, solution cannot be optimized further or maximum iteration of CSA have been performed.

Algorithm 2:

1. Select a program or software to be tested or for whose test cases need to generate.
2. Initialize variables like population_size, step_size, max_generation, generation_counter, total_targets, target_achieved_counter
3. Select a suitable software testing parameter like path coverage, node coverage, decision coverage, du-path coverage for testing adequacy criterion and find out the total number of the target to be achieved then assigned this value to total_targets
4. Generate an initial population (ie test cases) randomly
5. Check how many targets are achieved using the initial population and change the value of total_targets appropriately, if required
6. **For** each uncovered target
7. **While** generation_counter is less than max_generation
8. Select a new uncovered target from the target list
9. Run test cases one by one and calculate its fitness value using objective function and save it
10. **If** the target is reached **then**
11. Increase target_achieved_counter and break the while loop
12. **end if**
13. Randomly generate a test case within input domain (say cuckoo_solution) and run the program using this input and calculate fitness value of this test
14. Randomly select a test case from the current population and compare it with cuckoo_solution using its fitness value and do the following
15. **If** cuckoo_solution has better fitness value **then**
16. Replace selected test case with cuckoo_solution
17. **else** discard the cuckoo_solution
18. Remove worst solution (test case with least fitness value) from the population
19. Select an input test case from the current population, randomly
20. Perform Le´vy Flight on it and update its value in the current population
21. Increment the value of generation_counter
22. **end if**
23. **If** the target is reached **then**
24. increase target_achieved_counter and break the while loop
25. **end if**
26. **end while**
27. **if** generation_counter is equal to max_generation **then**
28. Declare remaining targets as unreachable and break the for loop
29. **end if**
30. **end for**
31. **If** all targets are reached then exit

V. CONCLUSION AND FUTURE WORK

After doing the literature survey meticulously and research, in the field of optimization, using nature-inspired algorithms. This paper proposed a framework for optimization of test case generation using Cuckoo Search Algorithm. This framework may be useful for the software engineering fraternity. The proposed framework will be implemented on various benchmark programs, which will be helpful for the generation of optimal test cases.

REFERENCES

[1] McMinn, P. “Search-based software testing: Past, present and future”. In Software testing, verification andvalidation workshops (icstw), IEEE fourth international,conference on IEEE. 2011, pp. 153-163.

[2] McMinn, P. “Search-based software test data generation: a survey”. Software testing, Verification, and reliability, 2004, pp.105-156.

[3] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, 2009, pp. 210-214.

[4] Yang, X. S., and Deb, S. “Engineering optimization by cuckoo search”. International Journal of Mathematical Modelling and Numerical Optimisation, 2010, pp 330-343.

[5] Valian, E., Mohanna, S., and Tavakoli, S. “Improved cuckoo search algorithm for feedforward neural network training”. International Journal of Artificial Intelligence & applications, 2011, pp. 36-43.

[6] Chandrasekaran, K., and Simon, S. P. “Multi-objective scheduling problem: a hybrid approach using fuzzy assisted cuckoo search algorithm”. Swarm and Evolutionary Computation, 5, 2012, pp. 1-16.

[7] Gandomi, A. H., Talatahari, S., Yang, X. S., and Deb, S. 2013. “Design optimization of truss structures using cuckoo search algorithm”. The Structural Design of Tall and Special Buildings, 2013, pp.1330-1349

[8] Srivastava, P.R., Khandelwal, R., Khandelwal, S., Kumar, S., Ranganatha, S.S., “Automated test data generation using cuckoo search and tabu search (csts) algorithm”. J. Intell. Syst. 21(2), 2012, pp.195–224

[9] Sur, C., and Shukla, A., “Discrete Cuckoo Search Optimization Algorithm for Combinatorial Optimization of Vehicle Route in Graph-Based Road Network”. In Proceedings of the Third International Conference on Soft Computing for Problem Solving, 2014, pp. 307-320.

[10] Jeya Mala Dharmalingam, Sabarinathan K and Balamurugan S, “A Hybrid Test Optimization Framework using Memetic Algorithm with Cuckoo Flocking Based Search Approach”, ACM Proceedings of International Workshop on Search-Based Software Testing (SBST), Hyderabad, India, June 2–3, 2010, pp. 37–38

[11] Liang Z, Liao S, Wen Y, Liu X, Working parameter optimization of strengthen waterjet grinding with the orthogonal-experiment-design-based ANFIS. J Intell Manuf:,2016, pp 1–22

[12] B. Nasser, Y. A. Alsariera, A. R. A. AlSewari, K. Z. Zamli, "A cuckoo search based pairwise strategy for combinatorial testing problem", Journal of Theoretical & Applied Information Technology, vol. 82, 2015, pp. 154-162.

[13] Khari, M.; Kumar, P. “A novel approach for software test data generation using cuckoo algorithm”. In: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies. ACM. Mar 4, 2016, p 98

[14] K. Senthil Kumar, "Optimal test suite selection using improved cuckoo search algorithm based on extensive testing constraints" International Journal of Applied Engineering Research 12(9):, 2017, pp.1920-1928.

[15] Neenu George, K. Chandrasekaran, and A. Binu, Optimization-Aware Scheduling in Cloud Computing. In Proceedings of the International Conference on Informatics and Analytics (ICIA-16). ACM, New York, NY, USA, , Article 15 ,2016, pp. 1-5.

[16] Dharmendra Prasad Mahato, Cuckoo Search-Ant Colony Optimization Based Scheduling in Grid Computing. In Proceedings of the 47th International Conference on Parallel Processing Companion (ICPP '18). ACM, New York, NY, USA, Article 39, 10 pages, 2018, . DOI: https://doi.org/10.1145/3229710.3229750

[17] Xin Liu and Shenghua Dai, Optimization of Train Control Strategy for Energy Saving and Time Precision Using Multi-Objective Cuckoo Search Algorithm. In Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE '18). ACM, New York, NY, USA, Article 183, 5 pages. 2018, DOI: https://doi.org/10.1145/3207677.3278087

[18] Anna Syberfeldt and Simon Lidberg, Real-world simulation-based manufacturing optimization using Cuckoo search. In Proceedings of the Winter Simulation Conference (WSC '12). Winter Simulation Conference, 2012, pp. 1-12

[19] Yang XS, Deb S. “Multiobjective cuckoo search for design optimization”, Computers & Operations Research Volume 40, Issue 6, 2013, pp. 1616-1624

[20] Bhandari et al., “Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur’s entropy”, Expert Systems with Applications, 41 (7), 2014, pp.3538-3560

[21] A.R. Yildiz, “Cuckoo search algorithm for the selection of optimal machining parameters in milling operations”, International Journal of Advanced Manufacturing Technology,2013, pp. 55-61.

[22] Mahajan, M., Kumar, S., & Porwal, R., “Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach”, ACM SIGSOFT Software Engineering Notes, 37(5), 2012, pp. 1-5.

[23] Varshney, Sapna, and Monica Mehrotra, "Search-based test data generator for data-flow dependencies using dominance concepts, branch distance and elitism." Arabian Journal for Science and Engineering 41.3, 2016, pp. 853-881.

[24] Kumar, S., Yadav, D. K., & Khan, D. A., “A novel approach to automate test data generation for data flow testing based on hybrid adaptive PSO-GA algorithm”, International Journal of Advanced Intelligence Paradigms, 9(2-3), 2017, pp. 278-312.

[25] Kumar, Sumit, D. K. Yadav, and D. A. Khan. "An accelerating PSO algorithm based test data generator for data-flow dependencies using dominance concepts."International Journal of System Assurance Engineering and Management8.2, 2017, pp. 1534-1552.