Research on use of Nature Inspired Algorithms in Software Testing

Sanjiv Sharma, S.A.M. Rizvi, Vineet Kumar Sharma

Abstract—The intention behind software testing is discovering defects in the developed software, making it error-free, robust and trustworthy. A large quantum of total efforts in the development of software is done on software testing. These efforts may include time, cost and most importantly manpower. The number of efforts required in software testing may be reduced if test data is generated automatically, without trade off the quality of the developed software. In literature, various nature inspired algorithms are used for the optimization of the software testing process. This article is a brief study on the applications of nature inspired algorithms in software testing. To keep the study succinct this study considers only two widely used testing types, structural and functional testing and literature available, since 2010 by considering only SCI or SCOPUS indexed publications.

Index Terms—structural testing, functional testing, software testing, test data generation, nature inspired algorithm, testing optimization

1. INTRODUCTION

Software testing is among the important phases in the software development cycle. The sole aim of this phase is to detect faults in the developed software and make it error-free, robust and trustworthy. A large quantum of total efforts in the development of software is done on software testing. These efforts may include time, cost and most importantly manpower. In software testing phase first step is to generate test cases, to works as an input for the developed software and then it is checked that whether the software under test is meeting its requirement or missing something. To make developed software error free exhaustive testing needed, which in turn requires a lot of efforts, so it is a challenge for someone to do it optimally. The number of efforts required in software testing may be reduced if test data is generated automatically, without any adverse effect on the quality of the developed software. Test case generation is an optimization problem and can be solved optimally with the help of nature-inspired optimization algorithms. These nature-inspired algorithms may include Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Cuckoo Search Algorithm (CSA), Firefly Algorithm (FA), etc.

This article presents a brief study on the applications of the nature-inspired algorithm in software testing since 2010 while considering only SCOPUS or SCI-indexed research papers from online digital libraries (ACM, IEEE, Springer, ScienceDirect and, Elsevier). Fig. 1 shows the number of publications year wise and Fig. 2 shows the shares of SCI and SCOPUS indexed research papers and Fig. 3 shows shares of Conferences and journals. In literature there are various nature inspired algorithms, this study considers only GA, PSO, ACO, CSA and FA because these algorithms are widely used. Relative contributions of these algorithms are shown in Fig. 4. To keep the study concise this article categorizes available literature in two categories of software testing, i.e. structural and functional testing. This paper is composed as follows, section II gives the overview of GA and its use in software testing, section III summarizes PSO algorithm and its application in software testing. Section IV, V and VI gives brief introductions of ACO, CSA and FA and their uses in software testing respectively. Finally, section VII represents the conclusion of this study.

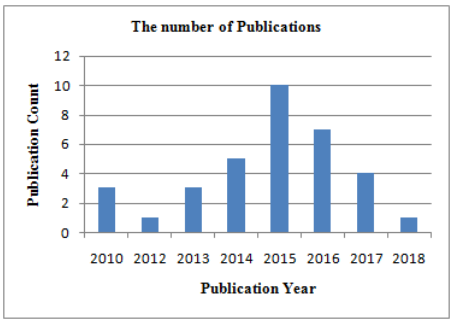

Fig 1: Publication from 2010 to 2018

2. GENETIC ALGORITHM

Genetic Algorithm (GA) was developed by John Holland [1] and it is the most widely used meta-heuristic based optimization algorithm. This algorithm is developed by taking inspiration from the theory of natural evaluation given by the great biologist Charles Darwin and it mimics the process of natural selection, where fittest or best individuals are selected by the nature for the breeding. GA is generally used to generate better solutions for the search problems having very large state space and optimization problem by using operations, such as crossover, mutation, and selection. A set of possible solutions to the problem is represented as a set of the chromosome. A set of chromosomes is named as population. A chromosome is represented as a string of character/binary digit/integer. Each individual chromosome denotes a solution to the problem or a point in the search space. GA uses the following rules in each iteration, to create the next generation from the current population.

Revised Version Manuscript Received on September 09, 2019.

Sanjiv Sharma, Computer Science & Engineering, KIET Group of Institutions, Ghaziabad, India

S.A.M Rizvi, Computer Science, Jamia Millia Islamia University, New Delhi, India

Vineet Kumar Sharma, Computer Science & Engineering, KIET Group of Institutions, Ghaziabad, India

a) **Initial population:** A set of initial solution is generated using randomly selected solutions. This set is represented as initial population.

b) **Fitness function:** This function is used to calculate quality of the current solution (chromosome) and assign a fitness score to it. This fitness score plays an important role in the selection of an individual for reproduction. Calculation of fitness function is totally dependent on the type of problem in hand and varies problem to problem.

c) **Selection:** This operation is used to select chromosomes that fulfill the fitness criteria, for reproduction and to generate a new population for the next iteration. Selection method may be one from roulette wheel, binary tournament, stochastic sampling, and truncation.

d) **Crossover or recombination:** This stage is used after selection and applied to selected chromosomes in the previous stage. In crossover two individual swap genes or sequence of bits on crossover sites. It may be on single point, double points, uniform or random. This stage creates a new offspring for the next iteration.

e) **Mutation:** This is used to keep diversity in the population of genes, some of the genes are mutated by altering their values. This process removes the premature convergence. Some important mutation operators are bit string, bit flip, boundary, uniform, and Gaussian.

A. GA in Structural Testing

A modified GA is used by [2] for automatic test case generation. The suggested approach uses real numbers instead of binary numbers for improvement of genetic algorithm and claims that this modified approach provides a better path and branch coverage for Delaunay Triangulation network program but has poor time efficiency compare to basic genetic algorithm. Vivanti et al. [3] used a search-based approach GA with the help of EVOSUITE tool for finding out test cases for data flow testing. Using EVOSUITE an empirical study is performed on the 100 open source java projects, selected from SF100 corpus and concludes that data flow testing provides better coverage than branch coverage and better mutation score. Garg and Garg [4] suggested a hybrid GA called HGA to generate test cases while considering a fitness function based on branch coverage. This new suggested hybrid algorithm is a blend of hill climbing and a simple genetic algorithm. Experiment results show that suggested approach performs better than simple path coverage. Yang et al. [5] used a modified GA called regenerate genetic algorithm (RGA), which solves the population aging problem of the basic genetic algorithm. This algorithm regenerates population when population aging crosses the threshold limit. For experiment Siemens suite's programs are selected for evaluation of the suggested approach. In this work, results are compared with the basic Genetic Algorithm and random testing by considering branch coverage as a test adequacy criterion. Khan and Amjad [6] used GA and mutation analysis for generation of test cases of the selected program while considering path coverage and boundary coverage as testing adequacy criterion. As per the study claim, this approach achieved 100 % path and boundary coverage. Varshney and Mehrotra [7] suggested GA based approach for generating optimized test cases for data flow testing while considering all uses criterion as testing adequacy criterion. This paper also introduces a fitness function by using dominance concept in control flow graph, elitism and branch distance. Results achieved using this approach are also compared with random testing and the results produced by the [8].

B. GA in Functional Testing

A combination of GA and static analysis for automatic testing of Eiffel classes is used in [9]. In this work, the fitness function depends upon the number of faults identified by test cases and results are compared with random testing. This study claims that the suggested approach performs better than random testing for the object-oriented software. GA is used by [10] in black box testing. In this work test cases are generated by considering the use case diagram as an input program. The fitness function is the ratio between valid and invalid test results. Arora [11] used GA with variable length chromosome, for the generation of test suite in the state-based testing. In this study, a program is represented as a fine state machine and fitness function depends upon the number of states traversed and transitions. Betts and Petty [12]used GA for automatic testing of UAV flight control software. To achieve optimized test cases fitness function depends upon maximum lateral deviation from the predetermined path is used. This study claims that GA and surrogate-based optimization algorithms perform better than the commonly used Monte Carlo testing method.

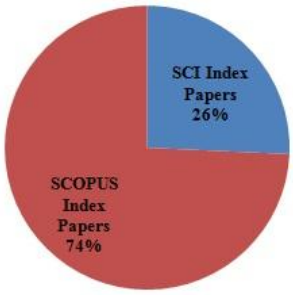

Fig 2: Share of SCOPUS and SCI indexed Papers

III. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) algorithm is also nature inspired optimization algorithm. This is also population-base optimization algorithm developed by Kennedy and Albert in 1995 [13]. It mimics the behavior of bird flocking and schooling pattern of fish. PSO uses a set of particles called swarm and this swarm moves around in the search space and explore possible solutions.

Movements of any particle is guided by the following three things.

a) Particle's current velocity.
b) Distance from its best-known position.
c) Distance from the swarms best position.

All particles of swarm communicate with each other and share their finding in the search space. Movement or trajectory of particle has two components i.e. deterministic and stochastic. Each particle's position is defined using its position vector (X_i) and velocity vector (V_i). For a d dimensional space, each particle is represented as a d-dimensional positional vector $X_i = [X_{i1}, X_{i2},\ldots\ldots, X_{id}]$ and all particle collectively called population. The previous best value achieved by a particle is called P_{best} and described as $P_{besti} = [P_{besti1}, P_{besti2},\ldots\ldots, P_{bestid}]$. The best value of each particle is called global best (P_{gbest}) and described as $P_{gbesti} = [P_{gbesti1}, P_{gbesti2},\ldots, P_{gbesti1d}]$. Velocity of the particle and described as $V_i = [V_{i1}, V_{i2},\ldots, V_{id}]$. Velocity of i^{th} particle in j^{th} iteration is calculated using the following equation.

$$V_{id}(j+1) = WV_{id}(j)+C_1R_1(P_{besti}(j)- X_i(j))+C_2R_2(P_{gbesti}(j)- X_i(j)) \quad (1)$$

Where value of i varies from 1 to n and n signifies population of the swarm, w is used to denotes inertia weight, c_1, and c_2 denotes acceleration constants, r_1 and r_2 are two randomly generated number in the range [0,1]. Equation 2 is used to calculate position of the i^{th} particle in the j^{th} iteration.

$$X_i(j+1) = X_i(j)+ V_{id}(j+1) \quad (2)$$

Based on the above two equations the population of particles converges on a global best solution while each particle is having random movement.

A. PSO in Structural Testing

Nayak and Mohapatra [14] used PSO for the generation of test suite for structural testing using data flow testing. This new approach is tested on the 14 FORTRAN programs while considering all uses criterion as test adequacy criteria. The results achieved after simulation of the suggested approach on MATLAB are compared with the solution achieved using GA algorithm. Using an empirical study of results achieved after experiments, a conclusion is derived that PSO performs much better than GA in achieving 100% def-use coverage. In other words, PSO achieved the same results in lesser number of generation compared to GA. Agarwal and Srivastava [15]used a modified PSO algorithm, called Discrete Quantum Particle Swarm Optimization (QPSO), to generate test cases automatically for three benchmark programs e.g. triangle classifier, calculation of number of days between two given dates and line in a rectangle problem. The branch coverage is considered as test adequacy criteria and fitness function is also based on branch predicates in branching condition. Mao [16] also used PSO for the generation of test suite while considering a objective function depending on branch coverage and branch distance for the guidance of the PSO in its search space. An empirical study is performed after finding test cases using PSO for eight benchmark programs and test cases produced by Simulated Annealing (SA) and Genetic algorithm on matrices like average percentage coverage, success rate, average generations and average execution time. In the end, this study concludes that PSO performs better than SA and GA. Jiang *et al.* [17] used a modified PSO algorithm called Reduced Adaptive PSO, for automatic test data generation. This suggested algorithm modifies the evolution equation after removing the velocity component of PSO and considers only inertia weight. This modification in the PSO reduces the chances of getting stuck in local minima while searching in the state space. The inertia weight suggested in this work is dynamic and changes its value based on the relationship between particle fitness and aggregation degree. For guidance to the suggested algorithm, a fitness function is used. This fitness function works on branch predicates. After each execution of the algorithm updated population is partitioned into three parts and the inertia of each part is calculated independently. As per the study, this provides a better balance between local and global search. For the validation of the suggested work, this new approach is applied to four benchmark program as well as four industry programs and compares the results with two other approaches suggested by [18] [19]. After experimentation of suggested work, it is concluded that this new approach provides better convergence speed. Sumit [20] used a hybrid approach for the generation of test suite in data flow testing. This hybrid algorithm is called Adaptive PSO-GA, and it is an amalgamation of GA and PSO. This algorithm removes the problem of immature convergence from PSO and the problem of slow convergence of GA. In this study, a new objective function is also suggested for the guidance of the adaptive PSO-GA. This objective function is a combination of dominance relation, branch weight, and branch distance in a CFG of the program under test. The effectiveness of this new approach is tested on four real-world problems as well as on the ten benchmark programs and further results are compared with results gained using Differential Evolution (DE), PSO, GA, and ACO on two parameters i.e. an average number of generation, average coverage achieved. In the end, the study claims that the suggested approach produced a better result than above-mentioned algorithms. Kumar, Yadav, and Khan [21]suggested a modified PSO called accelerating PSO for data flow testing. This suggested ASPO provides a better tunning between exploration and exploitation. This work also introduces a new objective function based upon the dominance relation between nodes of the CFG and branch weight distance for the guidance of ASPO in its search space. A set of 10 benchmark programs are used for the empirical study of the suggested approach and the study concludes that the suggested approach works better than random search, GA and PSO algorithm for the same set of programs on several parameters like the number of generation, average coverage, and ANOVA test. Varshney and Mehrotra [22]used a combination of PSO and differential Evolution to generate test data for structural testing. This suggested approach is used for data flow testing with the help of neighborhood search strategy for improvement in the performance of the suggested hybrid algorithm. The fitness function used in this work is based upon dominance concepts of nodes in CFG of the program under test and branch distance, while considering the mean number of generation and percentage of coverage as performance matrices for evaluation of the suggested hybrid algorithm. The results achieved by this algorithm are compared with results gain from using GA, Random Search, PSO and Differential Evolution algorithms

on 10 benchmark programs and results indicate that this new approach performs better than others on the selected benchmark programs.

B. PSO in Functional Testing & Results

A simplified version of PSO called Simplified Swarm Optimization (SSO) is used by Ahmed, Sahib, and Potrus [23] for GUI functional testing. This simplified algorithm SSO modifies the velocity modification equation by removing personal influence. After generating the final set of test cases an empirical study is done by comparing results of SSO with Test Vector Generator (TVG), Pairwise Independendant Combinatorial Testing (PICT), Intelligent Test Case Handler (ITCH) and Parameter Order Generator (IPOG) with the help of Quick Test Professional (QTP), an automatic testing software. For the case study, example software for fight reservation written in Visual Basic is used. As per the study's claim, the suggested strategy generates a lesser number of test cases than the original PSO version. Tyagi and Malhotra [24] used Multi-Objective PSO in regression testing for prioritization of test cases. This suggested approach works in three steps. In the first step, matrix operations are used to remove redundant test cases. In the second step, PSO is used for finding out minimal test suite which covers all faults in minimum execution time and priority is assigned to the test cases as a last step. For experiment two case studies are used and results are generated after simulation on MATLAB. The results achieved using the suggested algorithm, are compared with three other approaches of regression testing called No Ordering, Reverse Ordering and Random Ordering. In the end, it is concluded that the suggested approach works better than the rest three one.

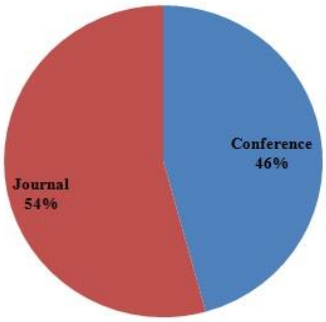

Fig 3: Share of Conferences and Journals

IV. ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) is another nature-inspired optimization algorithm that mimics the foraging behaviour of an ant colony. This algorithm is suggested by Dorigo [25]. During foraging process ant drops the special chemical called pheromone on the traversed path. This pheromone fades away over time. When other ants follow the same path, they also deposit the pheromone on the traversed path. In the initial phase, ants select the random paths for searching food and deposit the pheromone while moving. When some ant finds out shortest route form colony to food it deposits pheromone on the path of it's to and fro route, this activity leads to the high intensity of pheromone on that path which further attracts other ants to follow the same path. After some tine, all ants converge on that shortest path.

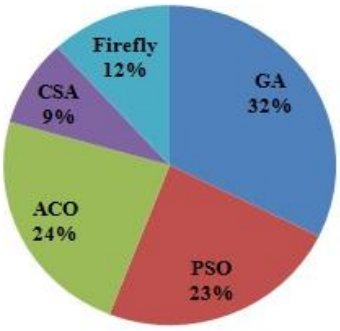

Fig 4: Relative Contribution of Algorithms

A. ACO in Structural Testing

Mao, Xiao, Yu, & Chen [26] used a discrete version of the ACO algorithm to generate test data for structural testing while taking branch coverage as test adequacy criteria. This work proposes a new fitness function, based on nesting level and predicate coverage of the branch. The suggested approach is tested on eight benchmark programs and results are compared with simulated annealing and genetic algorithm. As per the study claim suggested approach outperforms the other algorithm mentioned above in terms of coverage capability and convergence speed. Yang, Man, and Xu [27]used modified ACO for software test case generation. This modified ACO introduces a new coefficient with the name Improved Pheromone Volatilization Coefficient for ACO (IPVACO) for pheromone update strategy. The results achieved using this approach are compared with GA and Random Approach by considering branch coverage and statement coverage as testing adequacy criterion. After the experiment, an empirical study is used to establish that approach based on IPVACO is better than the others. Biswas, Kaiser, and Mamun [28] used ACO algorithm in structural testing. This suggested approach used to generate test suite for path coverage and then prioritizes them. Although this approach is used on a single program i.e. binary search and gives evidence that by using this approach it is guaranteed that with the help of this full software coverage can be achieved without having any redundancy.

B. ACO in Functional Testing

Noguchi and Washizaki [29] used ACO in black box testing and prioritizes the test cases. A framework is suggested for the prioritization of test suite on a new product, with the help of test execution history of similar older product. For experiment two actual products are considered. One is software for medical purpose and another one is for financial purpose. After simulations, results achieved using the suggested framework are compared with random order approach while considering the Average of the Percentage of Faults Detected (APFD) as an evaluation parameter. In the end, it is established that, suggested approach provides better APFD than the random approach. Zheng and Hu [30]used a hybrid version of ACO for the creation of automated test data sequence for the functional requirement specification of a

high-speed train in China. This hybrid ACO is the combination of ACO and Maze algorithm. This new algorithm provides dynamic learning capability and overcomes the problem of local optima of plain ACO algorithm. For the experiment purpose, four scenarios are used after converting them into Colored Petri Nets (CPN) and then converted into an XML model. At the last step modified ACO is used to generate optimal sequence of test cases. The generated sequence further tested on the system after creating the Radio Blocking Center (RBC) test platform. The results achieved after this indicates that this new approach optimizes the test sequences. Dongdong Gao, Xiangying Guo [31] used ACO for in the regression testing and prioritizes the test cases. This approach optimizes the test sequence after considering three factors; the number of faults detected the severity of fault and execution time of the program as a guiding tool in the search state space. The results achieved using this, are compared with the random approach by considering APFD as a metric. Sayyari, Faeghe [32] used ACO on model-based testing. In this approach, a small model of phone is considered and control flow graph of phone communication is created while assigning different weight to the branches. ACO with Markov chain is used for finding out optimal test paths and puts a limit on the generation of paths with p-factor. Carino and Andrews [33] used ACO for dynamic GUI testing. In this approach, two variant of ACO are used. One is simple ACO and other is AntQ. The AntQ algorithm is a hybrid of ACO and Q-Learning (a behavioral reinforcement technique). Both algorithms are used for traversing the GUI state diagram and finding out good event sequence while using the same fitness function. Here fitness function is maximization function and finds out the amount of change in the GUI state for every test case. The test cases which have a higher impact on the GUI state carry forward to the next iteration. Results achieved using the above two approaches are compared with results achieved using random selection technique by using six applications and it is concluded that AntQ achieved the highest code coverage.

V. CUCKOO SEARCH ALGORITHM

Xin-She-Yang and Suresh Deb in 2009 [34] developed Cuckoo Search Algorithm (CSA). CSA mimics the behavior of some cuckoo species, which have parasitic nature for their brooding. These cuckoos lay their eggs in such nests in which eggs are laid recently by the host bird and have resemblance with cuckoo eggs. If host bird somehow figures out that all eggs do not belong to them, in such case it has two choices. First, it may push the alien eggs out from nest and second, it may simply abandon the nest and make new nest somewhere else. In most of the cases, cuckoo's eggs hatch earlier than host bird's eggs and once cuckoo chick hatched, it removes the host egg from the nest by following its natural instinct. This action of cuckoo chick improvises the probability of its survival by accessing more food brought by the host the host bird. To mimic the process of searching a host bird by a cuckoo, CSA uses the concept of Le´vy flight. Le´vy flight is used to exhibit the foraging behavior of various animal and insects. CSA works on the following three principles.

a) One cuckoo chooses a nest randomly from a finite set and lay single egg in it at a time.

b) The nest having high-quality eggs will be forwarded to the next generations.

c) Availability of host nests is fixed and egg laid down by cuckoo may be identified by host bird by the probability P_a.

A. CSA in Structural Testing

Panda, Sarangi, & Dash [35] used CSA in unit testing while considering all feasible path coverage of a CFG as test adequacy criteria. The objective function used in this research article is based on edge weight based path coverage. The suggested approach has been applied on a single benchmark program i.e. triangle classifier program and results are compared with PSO and Gravitational Search Algorithm (GSA). After comparisons of results, it is established that the suggested approach is superior to PSO and GSA. Khari Manju[36] applied CSA in the structural testing. The test adequacy criterion considered in this work is path coverage. The suggested approach is tested on twenty benchmark programs and the results achieved are compared with the Hill Climbing Algorithm using three comparison parameters. These parameters are the size of the optimized test data, number of iterations and duration of execution time. As per the study claim, results achieved using CSA are better than Hill Climbing Algorithm. Sharma, Rizvi, and Sharma[37] suggested a framework and algorithm for the functional testing. In this work a CSA bases algorithm is proposed to generate test cases.

B. CSA in Functional Testing

Nagar, Kumar, Singh, & Kumar [38] used CSA in regression testing. This approach prioritizes the test suite from a given test case pool on the basis of the number of faults identified in minimum time. This approach is tested on a random case study and simulation is done using MATLAB. From the achieved results it is concluded that CSA reduced the test suite by 40%.

VI. FIREFLY ALGORITHM

Firefly algorithm (FA) is another nature inspired optimization algorithm developed by Xin She Yang [39]. This algorithm is based upon the social flashing behavior of fireflies and mimics the behavior of firefly for finding mates, attracting its prey and protecting themselves from predators using its flashlight. This algorithm is multi-objective optimization algorithm and works on the following three principles.

a) All fireflies are unisex and any firefly can attract other fireflies regardless of its sex.

b) The attractiveness of firefly is directly propositional to its brightness and both depend on the distance. If distance increases then both of them decreases and vice versa. For any tow arbitrary fireflies, the firefly having lesser brightness moves towards the firefly having higher brightness level. If all flies have same level of brightness level, firefly moves randomly.

c) Brightness of flashing light is determined by the objective function of the problem concerned and in most of the cases it is maximization function.

A. FA in Structural Testing

Pandey & Banerjee [40] used a modified version of Firefly algorithm called chaotic firefly, to generate test cases for white box testing. The suggested approach is applied on five benchmark program while considering path coverage as testing adequacy criteria. The results achieved using the firefly algorithm are compared with 3 other meta heuristic based optimization algorithm ACO, ABC and GA while considering path coverage and execution time as a comparison parameter. After simulation on MATLAB, the results achieved, indicates that chaotic firefly algorithm outperforms the rest.

B. FA in Functional Testing

Panthi & Mohapatra [41] have applied Firefly Algorithm (FA) for the generation of a test data sequence of software, using UML modeling. In this approach software is converted into a state machine diagram, the Firefly algorithm is used for the generation of prioritized sequence for the state machine. For the validation of the suggested approach bank ATM system is considered as a case study. Srivatsava, Mallikarjun, and Yang [42] used Firefly algorithm for optimal test case sequence generation for software testing and developed a tool "Optimal Firefly Test Sequence Generator (OFTSG)". To apply this suggested approach a program is represented as a graph using State transition Diagram and CFG. For experiment a single case study is taken and results are compared with results achieved using ACO, after simulation of the suggested approach. After doing empirical study it is found that firefly, generates less redundant test cases as compared to ACO. Sharma & Saha [43] applied Firefly algorithm in model based testing. In this suggested approach a program is represented as a state transition diagram, based on its behavior on inputs. For the validation of suggested work, it is applied to the five benchmark programs and test cases are generated using Firefly algorithm. After that, a comparative study is done by comparing the results achieved using FA and ACO. From the empirical study, it is established that results achieved using FA are less redundant than ACO.

CONCLUSION

This article presents a study on the use of nature inspired optimization algorithms in software testing optimization since 2010. After the study of the literature available from authentic and reputed sources (i.e. SCOPUS and SCI indexed publications), the following conclusions have been drawn. First is that GA is the most widely used optimization algorithm while CSA and Firefly algorithms are the least used ones. Second, most of the work done is in the white box testing especially, structural testing. The third is, in structural testing, path and branch coverage are the most extensively used as test adequacy criteria. And last observation is, the majority of the work is validated on some small benchmarked program e.g. Triangle classifier program, checking number is a perfect square or not, etc. From the above observations, we can say that use of nature inspired algorithm in the field of testing is in the growing phase, there is ample scope for work in this research area.

REFERENCES

1. J. H. Holland, *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press, 1992.
2. D. Liu, X. Wang, and J. Wang, "Automatic test case generation based on genetic algorithm," *J. Theor. Appl. Inf. Technol.*, vol. 48, no. 1, pp. 411–416, 2013.
3. M. Vivanti, A. Mis, A. Gorla, and G. Fraser, "Search-based Data-flow Test Generation," in *IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, 2013, pp. 370–379.
4. D. Garg and P. Garg, "Basis Path Testing Using SGA & HGA with ExLB Fitness Function," *Procedia Comput. Sci.*, vol. 70, pp. 593–602, 2015.
5. S. Yang, T. Man, J. Xu, F. Zeng, and K. Li, "RGA: A lightweight and effective regeneration genetic algorithm for coverage-oriented software test data generation," *Inf. Softw. Technol.*, vol. 76, pp. 19–30, 2016.
6. R. Khan and M. Amjad, "Optimize the Software Testing Efficiency using Genetic Algorithm and Mutation Analysis," *2016 3rd Int. Conf. Comput. Sustain. Glob. Dev.*, vol. 16, pp. 1174–1176, 2016.
7. S. Varshney and M. Mehrotra, "Search-Based Test Data Generator for Data-Flow Dependencies Using Dominance Concepts, Branch Distance and Elitism," *Arab. J. Sci. Eng.*, vol. 41, no. 3, pp. 853–881, 2016.
8. A. S. Ghiduk, M. J. Harrold, and M. R. Girgis, "Using genetic algorithms to aid test-data generation for data-flow coverage," *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, pp. 41–48, 2007.
9. L. S. Silva and M. van Someren, "Evolutionary testing of object-oriented software," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, p. 1126.
10. M. Fischer and R. Tonjes, "Generating test data for black-box testing using genetic algorithms," in *IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, 2012.
11. A. Arora and M. Sinha, "State based test case generation using VCL-GA," *Proc. 2014 Int. Conf. Issues Challenges Intell. Comput. Tech. ICICT 2014*, pp. 661–665, 2014.
12. [12] K. M. Betts and M. D. Petty, "Automated Search-Based Robustness Testing for Autonomous Vehicle Software," *Model. Simul. Eng.*, vol. 2016, pp. 1–15, 2016.
13. R. E. James Kennedy, "Particle Swarm Optimization," in *Proc. IEEE International Conf. on Neural Networks*, 1995.
14. N. Nayak and D. P. Mohapatra, "Automatic test data generation for data flow testing using particle swarm optimization," *Commun. Comput. Inf. Sci.*, vol. 95 CCIS, no. PART 2, pp. 1–12, 2010.
15. K. Agarwal and G. Srivastava, "Towards software test data generation using discrete quantum particle swarm optimization," in *Proceedings of the 3rd India software engineering conference*, 2010, pp. 65–68.
16. C. Mao, "Generating Test Data for Software Structural Testing Based on Particle Swarm Optimization," *Arab. J. Sci. Eng.*, vol. 39, no. 6, pp. 4593–4607, 2014.
17. S. Jiang, J. Shi, Y. Zhang, and H. Han, "Automatic test data generation based on reduced adaptive particle swarm optimization algorithm," *Neurocomputing*, vol. 158, pp. 109–116, 2015.

18. X. M. Zhu and X. F. Yang, "Software test data generation automatically based on improved adaptive particle swarm optimizer," in *International Conference on Computational and Information Sciences*, 2010, pp. 1300–1303.
19. S. Singla, D. Kumar, H. M. Rai, and P. Singla, "A hybrid PSO approach to automate test data generation for data flow coverage with dominance concepts," *Int. J. Adv. Sci. Technol.*, vol. 37, pp. 15–26, 2011.
20. S. Kumar, D. K. Yadav, and D. A. Khan, "A novel approach to automate test data generation for data flow testing based on hybrid adaptive PSO-GA algorithm," *Int. J. Adv. Intell. Paradig.*, vol. 9, no. 2/3, pp. 278–312, 2017.
21. S. Kumar, D. K. Yadav, and D. A. Khan, "An accelerating PSO algorithm based test data generator for data-flow dependencies using dominance concepts," *Int. J. Syst. Assur. Eng. Manag.*, vol. 8, no. s2, pp. 1534–1552, 2017.
22. S. Varshney and M. Mehrotra, "A Hybrid Particle Swarm Optimization and Differential Evolution based Test Data Generation Algorithm for Data-Flow Coverage using Neighbourhood Search Strategy," *Informatica*, vol. 42, no. 3, pp. 417–438, 2018.
23. B. S. Ahmed, M. A. Sahib, and M. Y. Potrus, "Generating combinatorial test cases using Simplified Swarm Optimization (SSO) algorithm for automated GUI functional testing," *Eng. Sci. Technol. an Int. J.*, vol. 17, no. 4, pp. 218–226, 2014.
24. M. Tyagi and S. Malhotra, "Test case prioritization using multi objective particle swarm optimizer," in *International Conference on Signal Propagation and Computer Technology, ICSPCT 2014*, 2014, pp. 390–395.
25. M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
26. C. Mao, L. Xiao, X. Yu, and J. Chen, "Adapting ant colony optimization to generate test data for software structural testing," *Swarm Evol. Comput.*, vol. 20, pp. 23–36, 2015.
27. S. Yang, T. Man, and J. Xu, "Improved Ant Algorithms for Software Testing Cases Generation," *Sci. World J.*, vol. 2014, p. 9, 2014.
28. S. Biswas, M. S. Kaiser, and S. A. Mamun, "Applying Ant Colony Optimization in software testing to generate prioritized optimal path and test data," *2nd Int. Conf. Electr. Eng. Inf. Commun. Technol. iCEEiCT 2015*, no. May, pp. 21–23, 2015.
29. T. Noguchi and H. Washizaki, "History-Based Test Case Prioritization for Black Box Testing using Ant Colony Optimization," in *IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015, pp. 2–3.
30. W. Zheng and N. W. Hu, "Automated test sequence optimization based on the maze algorithm and ant colony algorithm," *Int. J. Comput. Commun. Control*, vol. 10, no. 4, pp. 593–606, 2015.
31. L. Z. Dongdong Gao, Xiangying Guo, "Test Case Prioritization for Regression Testing Based on Ant Colony Optimization," 2015, no. 91118007.
32. F. Sayyari and S. Emadi, "Automated generation of software testing path based on ant colony," in *International Congress on Technology, Communication and Knowledge (ICTCK)*, 2016, pp. 435–440.
33. S. Carino and J. H. Andrews, "Dynamically testing GUIs using ant colony optimization," in *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 138–148.
34. X.-S. Yang and S. Deb, "Cuckoo Search via Levey Flights," *2009 World Congr. Nat. Biol. Inspired Comput. (NaBIC 2009)*, pp. 210–214, 2009.
35. M. Panda, P. P. Sarangi, and S. Dash, "Automatic Test Data Generation using Metaheuristic Cuckoo Search Algorithm," *Int. J. Knowl. Discov. Bioinforma.*, vol. 5, no. December, pp. 16–29, 2015.
36. M. Khari and P. Kumar, "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*, 2016, pp. 1–6.
37. S. Sharma, S. A. M. Rizvi, and V. Sharma, "A Framework for Optimization of Software Test Cases Generation using Cuckoo Search Algorithm," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2019, pp. 282–286.
38. R. Nagar, A. Kumar, G. P. Singh, and S. Kumar, "Test Case Selection and Prioritization using Cuckoos Search Algorithm," in *1st International Conference on Futuristic trend in Computational Analysis and Knowledge Management (ABLAZE-2015)*, 2015, pp. 283–288.
39. X. S. Yang, "Firefly algorithm, L??vy flights and global optimization," *Res. Dev. Intell. Syst. XXVI Inc. Appl. Innov. Intell. Syst. XVII*, pp. 209–218, 2010.
40. A. Pandey and S. Banerjee, "Test Suite Optimization Using Chaotic Firefly Algorithm in Software Testing," *Int. J. Appl. Metaheuristic Comput.*, vol. 8, no. 4, pp. 41–57, 2017.
41. H. S. Behera and D. P. Mohapatra, "Generating Prioritized Test Sequences Using Firefly Optimization Technique," *Adv. Intell. Syst. Comput.*, vol. 410, pp. 627–635, 2016.
42. P. R. Srivatsava, B. Mallikarjun, and X. S. Yang, "Optimal test sequence generation using firefly algorithm," *Swarm Evol. Comput.*, vol. 8, pp. 44–53, 2013.
43. R. Sharma and A. Saha, "Optimization of object-oriented testing using firefly algorithm," *J. Inf. Optim. Sci.*, vol. 38, no. 6, pp. 873–893, 2017.

AUTHORS PROFILE

Sanjiv Sharma is an assistant professor at KIET Group of Institutions, Ghaziabad. He received his B.Tech. degree from MMMEC, Gorakhpur affiliated to AKTU, Lucknow, India in 2008, M.Tech degree from Shobhit University Meerut, India in 2014, and pursuing Ph.D. from Jamia Milllia University, New Delhi, India. His research interests include software testing and nature inspired algorithm. One can connect Sanjiv Sharma on martin.mmmec@gmail.com.

S.A.M. Rizvi is a professor and former HoD at Jamia Millia University, New Delhi India. He received his Ph.D from Dr. R. M. L. Avadh University, India, in 1996. His research interests are Knowledge Engineering, MIS, Automation, Algorithms and Bioinformatics. One can connect SAM Rizvi on samsam_rizvi@yahoo.com.

Vineet Kumar Sharma is a professor and HoD at KIET Group of Institutions, Ghaziabad, India. He received his Ph.D. from Jamia Millia Islamia University, New Delhi, India in 2012. His research interest are algorithms, Software Engineering. One can connect Vineet Kumar Sharma on vineet.sharma@kiet.edu.