COMPUTER SCIENCE, COMPUTATIONAL METHODS, ALGORITHMS AND ARTIFICIAL INTELLIGENCE SESSION

NATURE-INSPIRED APPROACHES IN SOFTWARE TESTING OPTIMIZATION

Miodrag Živković[1]*,
Tamara Živković[2],
Nebojša Bačanin[1],
Ivana Strumberger[1]

[1]Faculty of Informatics and Computing,
Singidunum University,
Belgrade, Serbia

[2]School of Electrical Engineering,
Belgrade, Serbia

Abstract:
Software development is considered to be a fast growing industry that drives numerous modern world domains forward. At the same time, software testing and quality assurance, that is equally important branch of software industry, resides in the shadows, far away from the spotlights. The goal of the software testing is to detect defects in the software and ensure that it has sufficient quality prior to the release.

The main objective of the software testing can be defined as finding the minimal test suite that is still efficient enough to maintain the certain software quality. Since the process of test cases generation performs a search for an optimal test suite in a huge search space, and keeping in mind that the swarm intelligence metaheuristics have already proven to be efficient optimizers in other domains, it makes sense to apply swarm intelligence algorithms to the process of test cases generation as well. By utilizing this approach, it could be also possible to reduce the cost required for testing. This paper provides a survey of recent applications of swarm intelligence algorithms in the domain of software testing.

Keywords:
Cloud computing, information security, IT audit, compliance, ISMS.

INTRODUCTION

Software testing is a crucial activity that is often the decisive factor that determines if a software project will succeed or fail. The defects can be very expensive, especially if they were found late in the software development process. In this case, the process of debugging and fixing is costly, as the defect have probably propagated through multiple phases of the development, and affected other parts of software, documentation etc. Even worse, if defects haven't been found during testing process, they will typically be found by end-users in production, which will lead to financial loss, low customer satisfaction, poor reputation etc. Not to mention possible human casualties (medical applications or autonomous car software), substantial loss of client's data and money (banking applications)

Correspondence:
Miodrag Živković

e-mail:
mzivkovic@singidunum.ac.rs

DOI: 10.15308/Sinteza-2021-28-33

or leakage of client's private data such as documents, contacts, photos, call list and location (mobile applications).

Additionally, modern applications are complex, and the task to perform appropriate testing implies knowledge and mastery of different testing approaches. Most of the traditional testing techniques were introduced for simple procedural programs. However, the tester must apply different approaches to test object oriented software [1] or web applications [2].

Web applications are challenge on their own, as they are typically built in several different programming languages. Frontend is typically built with HTML, CSS and JavaScript, and it communicates with the backend through AJAX. Backend, on the other hand, can be built with either PHP, Node.JS, ASP .NET C#, or Java (JSP or Spring, for example). Finally, there is also the data persistence level, built with MySQL, Oracle, MongoDB or MSSQL. The options are numerous, and all parts are required to be tested thoroughly.

Besides the goals to find bugs in the software and ensuring that the software behaviour is correct according to the specification, software testing can be performed with a goal to build confidence in the application as well. The absence of critical defects will generally speaking increase the confidence in the software. However, one must be cautious as the actual absence of defects doesn't necessarily mean that the software is bug-free. One of the core principles of testing is that exhaustive testing is not possible for any non-trivial application, as the number of all possible combinations of input values is too large to be tested in a reasonable amount of time. In other words, the absence of defects just shows that the utilized test suite was not able to detect any bug. Therefore, the most important thing in software testing is to create a test suite that will cover all the functionalities of the application, with a special attention to the functionalities that are the most visible to the end-users (the functionalities that the users of the system will frequently use) and the use case scenarios (how the users will use the system).

According to the most estimations, software testing takes more than 50% of both the time and the cost of the software development process. This means that the test cases generation process is crucial, and if done properly, it can save the time and the cost of the entire project. Nature-inspired metaheuristics are proven to be exceptional optimizers in variety of other application domains, and it makes sense to assume that they can be applied in software testing as well, with a goal to optimize the test cases generation, and consequently to reduce the overall time and the cost of the complete project. This paper presents a survey of recent applications of the swarm intelligence and evolutionary algorithms in the software testing process optimization. The remainder of this paper is structured in the following way. Section 2 introduces the nature-inspired algorithms and gives the survey of the most important methods and their applications in various practical domains. Section 3 gives an overview of the practical implementations of the nature-inspired algorithms in the software testing domain. Finally, section 4 suggest the possible future work in this domain and concludes the paper.

2. SURVEY OF NATURE-INSPIRED APPROACHES

NP-hard challenges play a vital role in the modern computer science and have a significant practical importance in a large number of application domains, such as machine learning, wireless sensor networks and cloud systems. The common basis for all NP-hard problems is that it is not possible to solve them by applying the traditional deterministic approaches in an acceptable time-frame. They require another, stochastic approach, if the problem is to be solved in a reasonable time.

Metaheuristics methods belong to the group of stochastic algorithms. Their main purpose is to obtain the satisfactory solution to the problem (solution that is sufficiently good, but not imperatively the best one) in a predictable and reasonable time-frame [3]. The survey of the recently published works indicate that metaheuristics approaches have been used to solve a wide variety of different practical NP-hard challenges. One famous and important family of metaheuristics approaches are the nature-inspired algorithms. Nature-inspired metaheuristics can be roughly separated into two distinct sub-families: evolutionary algorithms (EA) and the swarm intelligence (SI) metaheuristics, respectively.

The EA methods are inspired by the natural selection and the premise that only the fittest individuals in a given population will survive. The fittest units are then chosen for breeding, and generating the offspring for the following generation that will inherit the favourable characteristics from the predecessors. After certain amount of iterations, the algorithm will produce the generation of the fittest units (solutions). The most famous representative of this group of algorithms is the genetic algorithm (GA) [4]. It was successfully applied in wide range of the practical NP-hard challenges, such as the load-balancing problem in the cloud-based systems [5], machine learning based Covid-19 prediction [6], design of the convolutional neural networks [7] and many others.

The latter family of the nature-inspired approaches, SI metaheuristics, was motivated by the behavior demonstrated by the groups of relatively simple animals, such as fireflies, bats, ants, moths, etc. Those simple animals can self-organize, show high level of coordination and perform complex actions when they form large groups (swarms). This particular feature of the swarms was the main source of inspiration for all SI algorithms [8]. Particle swarm optimization (PSO) was one of the earliest SI algorithms, and was proposed by Kennedy and Eberhart in 1995 [9]. It was inspired by the behavior demonstrated by the flocks of birds. Since its introduction, it was used to solve a variety of real-life problems, like cloud task scheduling [10]. Besides PSO, one more famous representative of the SI metaheuristics is the artificial bee colony (ABC) algorithm, inspired by the behavior of the honey bees in a hive, and introduced by Karaboga in 2007 [11]. ABC has also been used to solve a large number of different problems in various domains, for instance the portfolio problem as stated in [12]. Other famous SI approaches include the bat algorithm (BA) [13], firefly algorithm (FA) [14], and monarch butterfly optimization (MBO) [15] to name the few. Besides the mentioned metaheuristics approaches, there are dozens of other algorithms belonging to the SI family, with new approaches emerging every day.

Nature inspired approaches have recently been intensively used to address a wide spectrum of different NP-hard problems from a large number of application domains. SI metaheuristics were applied in the domain of the wireless sensor networks to solve the sensor node localization problem [16], prolonging the network lifetime [17], [18] and maximizing the network energy efficiency [19]. Another field where the SI metaheuristics have obtained respectable results is the cloud computing. SI was used to optimize the task scheduling by minimizing the overall time required to execute all tasks and the overall cost [20] [21] [22]. In the domain of machine learning, SI was utilized to design the convolutional neural networks [23], feed-forward neural network training [24] etc. Nature inspired metaheuristics also show very promising results in time-series prediction, the feature that was used to predict the Covid-19 cases [6] [25]. Finally, the SI approach was used to design a convolutional neural network that performs the classification task of the MRI images of the glioma tumor [26].

3. NATURE-INSPIRED APPROACH IN SOFTWARE TESTING

As discussed in the introduction, generating test cases and creating the appropriate test suite is crucial for the success of the project. The deficiencies in the testing process can be very costly and they will inevitably lead to defects occurring in the production. There are two contradictory requirements for a good test set. The first requirement is that it must be detailed enough to test all the important scenarios and paths through the program. However, the second requirement demands that the cost and time needed for testing are minimized as much as possible. It is clear that if we try to add more tests to the suite, we will increase the time and the cost of test execution. Vice versa, if we try to reduce the number of test cases with a goal to save time and money, it could result in the test suite which is not detailed enough and which will miss some defects.

Scientists in the software testing domain around the world have turned their attention to the artificial intelligence and nature-inspired approaches with a goal to address the test cases generation problem. As expected, the most popular nature-inspired metaheuristics were applied to this problem. The very first approach was to use GA to generate unit test cases automatically with a goal to obtain high code coverage. This approach is known as evolutionary structural testing (EST), and it was introduced all the way back in 1996 [27], and was later refined in 2006 [28]. EST has established to be efficient and successful for various academic test object instances, and also for some industrial projects. The main principle of EST is as follows: it is impossible to perform exhaustive testing, as the amount of time needed to test all the possible combinations of input values would be too long and impractical. Therefore, a subset of input data that is relevant must be selected. The term "relevant" here depends of the selected coverage criteria. Coverage criteria for the white box approaches are based on the control flow graph – CFG. For example, if a generated test set executes all the statements in the software component, it fulfills the statement coverage criterion. However, most of the industrial software must fulfill more strict criteria such as branch/decision coverage (all branches/decision outcomes in the CFG of the software component being tested must be exercised) or path coverage (all the independent paths through the CFG must be covered), which affects what is considered to be the "relevant" test set. The EST considers the test cases generation as the optimization problem that needs to be solved by using the search method, for instance, the GA. After selecting the test coverage criterion, the code is divided in the individual test goals which will be optimized separately. The fitness function is defined by utilizing the two distance metric: approximation level and branch distance, respectively. The former is related to the CFG of the software component that is being tested. More precisely, it correlates to the amount of critical branches, positioned between the problem and the target nodes in the graph. Here, the target node is the structure in the code that needs to be covered, while the problem node is the code structure where the program execution is diverging through a branch that will make reaching the target not possible. The latter, branch distance metric, corresponds to the condition found in the problem node. It serves to describe how close the condition has been evaluated to deliver the boolean result that was required to reach the target.

The original EST approach with GA implementation has been compared to the PSO implementation in [29]. The authors noted that since the PSO is easy to implement, efficient, and has the ability to converge fast to the optimal parts of the search space, it could also enhance the EST as well. They implemented both algorithms, and selected the branch coverage as the code coverage criterion. Furthermore, they considered each branch as an individual testing goal that needs to be optimized. They conducted the simulations over a total of 25 test object instances with different levels of complexity that have been created for the purpose of the experiment. The obtained results have proven that the PSO – based approach outperformed the GA in about 2/3 of the observed test instances. The authors concluded that the GA converges slightly faster if the functions are simple (in terms of parameters passed by), while the PSO drastically outperformed the GA in case of complex functions that have a large search space (mixed parameters, such as boolean, integer and/or double that are passed by when invoking the function). The final remarks of the paper indicate that the PSO is competitive with GA, and in cases of complex functions, it clearly outperforms the GA.

Paper [30] proposes the GA and PSO approach to the process of the test cases generation, by identifying the paths in software that are susceptible to defects. The proposed approach was named HGPSTA (hybrid genetic particle swarm technique algorithm), and it combines the individual advantages of GA and PSO. The authors have utilized the EST, that is used to generate the test cases automatically. EST considers this task as an optimization problem.

The GA has been hybridized with PSO with a goal to enhance the efficiency of the process. The GA utilizes three operators, namely selection, crossover and mutation. The GA hybridized with PSO uses enhancement operator, that is utilized to improve the units in the same generation. After calculation of the fitness value of all solutions of the population, the best half of solutions are noted. The algorithm then applies the PSO directly to improve the individual solutions. Crossover operator includes only the improved solutions, and the authors applied the roulette wheel scheme for the selecting process.

The proposed HGPSTA method was applied to the fitness function that utilizes the data flow testing coverage criteria, and tested on the seven classic programming problems. The HGPSTA was compared to the basic GA and basic PSO. Published results suggested that the HGPSTA approach was able to achieve 100% data flow coverage in less rounds than the basic GA and PSO algorithms, resulting in a smaller number of required test cases.

Another hybrid approach was proposed in [31], where the authors hybridized the GA with ACO, and named the approach hybrid ant colony genetic algorithm (HACGA). Their approach was based on the fault matrix, with 15 different defect and 15 test cases, with the assumption that every test case exposes at least one defect. The problem was further formulated as choosing the subset of the matrix's rows that covers each column (containing defects) at least once. The collection of test cases is chosen by the ACO search, and then refined by the GA. The conducted experiments included comparison with plain ACO and GA approaches as well. The proposed HACGA outperformed both GA and ACO in this particular test setup.

ACO – based approach was utilized as well in [32] with a goal to generate test data to cover the prime paths in the software. Prime paths in this context are the basic paths (no repetitions), and loops are tested just with 0 and 1 passes through the loop. Path coverage is considered to be the most strict criteria among white box techniques, as the 100% path coverage implies 100% decision coverage and 100% statement coverage as well. Therefore, covering the paths can discover defects that other techniques are not able to. The authors have tested their approach on seven benchmark programs with known bugs, and compared it to PSO and GA approaches. The obtained results suggested that ACO method outperforms the GA both in terms of coverage and efficiency. When compared to the PSO, authors noted that ACO obtains better test coverage than PSO, however, it is less efficient than PSO.

4. CONCLUSION

In this paper, we have performed a survey of nature-inspired metaheuristics that found their use in optimization of the software testing process. The hardest task in the software testing is the test cases generation process. The success of a software development project largely depends on it – if the test set is not adequate, it can lead to the complete failure of the project after the deployment. On the other hand, having a test set that has a large number of test cases is not efficient both in terms of the time and the cost required for the testing.

Nature-inspired metaheuristics have proven to be efficient optimizers. From the performed survey, it can be noted that there were several attempts to optimize the test generation by applying famous algorithms, including GA, PSO and ACO. The common goal for all mentioned approaches is that they try to generate test set that will increase the code coverage according to the selected criterion (statement, decision or path coverage), while decreasing the amount of test cases. It can be seen from the presented approaches that other researchers mostly play safe, by choosing and applying the traditional, famous algorithms. This leaves a lot of open space for applying modern, state-of-the-art metaheuristics (for example BA, FA or MBO), either in original, or their hybridized and/or improved versions to the same problem, with a reasonable chance to improve the test cases generation process even further.

Future research will focus on implementing one of the more recent metaheuristics to the software testing problem, and validating it against the traditional approaches such as GA and PSO executed on the same problem instance.

REFERENCES

[1] T. Živković and M. Živković, "Comparative Analysis of Techniques for Testing Object Oriented Programs," in *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, Novi Sad, Serbia, 2020.

[2] M. Živković and T. Živković, "Challenges in Testing of Web Applications," in *Sinteza 2018-International Scientific Conference on Information Technology and Data Related Research*, Belgrade, Serbia, 2018.

[3] I. Strumberger, N. Bacanin and M. Tuba, "Enhanced firefly algorithm for constrained numerical optimization," in *2017 IEEE congress on evolutionary computation (CEC),* San Sebastian, Spain, 2017.

[4] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[5] T. Wang, Z. Liu, Y. Chen, Y. Xu and X. Dai, "Load Balancing Task Scheduling Based on Genetic Algorithm in Cloud Computing," in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, Dalian, China, 2014.

[6] M. Zivkovic, K. Venkatachalam, N. Bacanin, A. Djordjevic, M. Antonijevic, I. Strumberger and T. A. Rashid, "Hybrid Genetic Algorithm and Machine Learning Method for COVID-19 Cases Prediction," in *Proceedings of International Conference on Sustainable Expert Systems: ICSES 2020*, Nepal, 2021.

[7] M. Suganuma, S. Shirakawa and T. Nagao, "A Genetic Programming Approach to Designing Convolutional Neural Network Architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Berlin, Germany, 2017.

[8] X. Yang, "Swarm intelligence based algorithms: a critical analysis," *Evolutionary Intelligence*, vol. 7, no. 1, pp. 17-28, 2014.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, IEEE, 1995.

[10] M. Kumar and S. Sharma, "PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 147-164, 2018.

[11] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459-471, 2007.

[12] M. Tuba and N. Bacanin, "Artificial bee colony algorithm hybridized with firefly metaheuristic for cardinality constrained mean-variance portfolio problem," *Applied Mathematics \& Information Sciences*, vol. 8, no. 6, pp. 2831-2844, 2014.

[13] X. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Springer, 2010.

[14] X. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International journal of bio-inspired computation*, vol. 2, no. 2, pp. 78-84, 2010.

[15] G. Wang, S. Deb and Z. Cui, "Monarch Butterfly Optimization," *Neural Computing and Applications*, pp. 1-20, 2015.

[16] N. Bacanin, E. Tuba, M. Zivkovic, I. Strumberger and M. Tuba, "Whale Optimization Algorithm with Exploratory Move for Wireless Sensor Networks Localization," in *International Conference on Hybrid Intelligent Systems,* 2019.

[17] M. Zivkovic, N. Bacanin, E. Tuba, I. Strumberger, T. Bezdan and M. Tuba, "Wireless Sensor Networks Life Time Optimization Based on the Improved Firefly Algorithm," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020.

[18] M. Zivkovic, T. Zivkovic, K. Venkatachalam and N. Bacanin, "Enhanced Dragonfly Algorithm Adapted for Wireless Sensor Network Lifetime Optimization," in *Data Intelligence and Cognitive Informatics*, 2021.

[19] M. Zivkovic, N. Bacanin, T. Zivkovic, I. Strumberger, E. Tuba and M. Tuba, "Enhanced Grey Wolf Algorithm for Energy Efficient Wireless Sensor Networks," in *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, Novi Sad, Serbia, 2020.

[20] T. Bezdan, M. Zivkovic, M. Antonijevic, T. Zivkovic and N. Bacanin, "Enhanced Flower Pollination Algorithm for Task Scheduling in Cloud Computing Environment," in *Machine Learning for Predictive Analysis*, 2020.

[21] T. Bezdan, M. Zivkovic, E. Tuba, I. Strumberger, N. Bacanin and M. Tuba, "Multi-objective Task Scheduling in Cloud Computing Environment by Hybridized Bat Algorithm," in *International Conference on Intelligent and Fuzzy Systems*, 2020.

[22] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba and M. Zivkovic, "Task scheduling in cloud computing environment by grey wolf optimizer," in *2019 27th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, 2019.

[23] I. Strumberger, E. Tuba, N. Bacanin, M. Zivkovic, M. Beko and M. Tuba, "Designing convolutional neural network architecture by the firefly algorithm," in *2019 International Young Engineers Forum (YEF-ECE)*, 2019.

[24] S. Milosevic, T. Bezdan, M. Zivkovic, N. Bacanin, I. Strumberger and M. Tuba, "Feed-Forward Neural Network Training by Hybrid Bat Algorithm," in *Modelling and Development of Intelligent Systems: 7th International Conference, MDIS 2020*, Sibiu, Romania, 2020.

[25] M. Zivkovic, N. Bacanin, K. Venkatachalam, A. Nayyar, A. Djordjevic, I. Strumberger and F. Al-Turjman, "COVID-19 cases prediction by using hybrid machine learning and beetle antennae search approach," *Sustainable Cities and Society*, vol. 66, 2021.

[26] T. Bezdan, M. Zivkovic, E. Tuba, I. Strumberger, N. Bacanin and M. Tuba, "Glioma Brain Tumor Grade Classification from MRI Using Convolutional Neural Networks Designed by Modified FA," in *International Conference on Intelligent and Fuzzy Systems,* 2020.

[27] B. F. Jones, H. Sthamer and D. E. Eyres, "Automatic test data generation using genetic algorithms," *Software Engineering Journal,* vol. 11, no. 5, pp. 299-306, 1996.

[28] J. Miller, M. Reformat and H. Zhang, "Automatic test data generation using genetic algorithm and program dependence graphs," *Information and Software Technology,* vol. 48, no. 7, pp. 586-605, 2006.

[29] A. Windisch, S. Wappler and J. Wegener, "Applying particle swarm optimization to software testing," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007.

[30] A. Singh, N. Garg and T. Saini, "A hybrid approach of genetic algorithm and particle swarm technique to software test case generation," *International Journal of Innovations in Engineering and Technology,* vol. 3, no. 4, pp. 208-214, 2014.

[31] P. Palak and P. Gulia, "Hybrid swarm and GA based approach for software test case selection," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 6, 2019.

[32] A. M. Bidgoli, H. Haghighi, T. Z. Nasab and H. Sabouri, "Using swarm intelligence to generate test data for covering prime paths," in *International Conference on Fundamentals of Software Engineering*, 2017.