A Hybrid Meta-Heuristic Approach for Test Case Prioritization and Optimization

Heba Mohammed Fadhil [*1], Mohammed Issam Younis [2]

[1] Department of Information and Communication, Al-Khwarizmi College of Engineering, University of Baghdad, Iraq
[2] Department of Computer Engineering, College of Engineering, University of Baghdad, Iraq
Emails: heba@kecbu.uobaghdad.edu.iq; younismi@coeng.uobaghdad.edu.iq

Abstract

The application of the test case prioritization method is a key part of system testing intended to think it through and sort out the issues early in the development stage. Traditional prioritization techniques frequently fail to take into account the complexities of big-scale test suites, growing systems and time constraints, therefore cannot fully fix this problem. The proposed study here will deal with a meta-heuristic hybrid method that focuses on addressing the challenges of the modern time. The strategy utilizes genetic algorithms alongside a black hole as a means to create a smooth tradeoff between exploring numerous possibilities and exploiting the best one. The proposed hybrid algorithm of genetic black hole (HGBH) uses the capabilities of considering the imperatives such as code coverage, fault finding rate and execution time from search algorithms in our hybrid approach to refine test cases considerations repetitively. The strategy accomplished this by putting experiments on a large-scale project of industrial software developed. The hybrid meta-heuristic technique ends up being better than the routine techniques. It helps in higher code coverage, which, in turn, enables to detect crucial defects at an early stage and also to allocate the testing resources in a better way. In particular, the best APFD value was 0.9321, which was achieved in 6 generations with 4.879 seconds the value to which the computer was run. Besides these, , the approach resulted in the mean value of APFD as 0.9247 and 0.9302 seconds which took from 10.509 seconds to 30.372 seconds. The carried out experiment proves the feasibility of this approach in implementing complex systems and consistently detecting the changes, enabling it to adapt to rapidly changing systems. In the end, this research provides us with a new hybrid meta-heuristic way of test case prioritization and optimization, which, in turn, helps to tackle the obstacles caused by large-scale test cases and constantly changing systems.

Keywords: Test Case Prioritization; genetic algorithms; black hole Algorithm; Software Testing; Meta-heuristic; Hybrid.

1. Introduction

The program of software test case prioritization having a tremendous effect on the productivity of software testing category and ensures software quality [1]. Software systems, due to their continuous improvement, become more complicated and limited resources that are allocated for testing turn into challenging tasks. In this regard, the enormousness and uneasiness of the current software cause the invention of novel assays for the mentioned administration [2], [3]. The test suites for large projects may be huge, usually containing many test cases and keeping constant pace with the continuous changes in the system. This causes test prioritization difficulties due to such complexities. This is about the ability to spot out crucial defects early during program testing and at the same time ensuring resource allocation is kept at a minimal level [4], [5].

Software testing has important role in maintaining and upholding the reliability and quality of software systems. Functioning as one of the crucial testing method, test case priority stays as the basic principle of optimizing the testing scope. Using test cases with the highest probability to pinpoint defects makes it possible for testing resources to be allocated more efficiently, resulting in enhanced fault detection rate, and this can be achieved by

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

using limited testing time and budget components [6]. Nevertheless, the estimation and solution of this optimization problem defy simple algorithms and require complex models to identify the optimum solution amidst frantic search in the vast solution space. Due to the increase in the complexity of the optimization processes, so as the hybrid metaheuristic approaches, in recent years built-up as ambitious options to overcome these issues [7], [8].

Genetic algorithms are evolutionary optimization concepts moulded from the process of natural selection. The algorithms repeatedly breed and change the population of solutions using genetic operators like crossover and mutation [9], [10]. GAs is proficient in a number of things such as exploring a diverse set of solutions; however, they may encounter difficulties in converging towards the optimum solutions easily [11], [12]. On the other hand, black hole optimization is a physics-inspired super-intelligent algorithm that intends to imitate the behaviour of black holes in space. Through applications of gravitational forces, black hole optimization techniques utilize promising areas in the solution space effectively and as result, the rate of convergence reduces to get to near optimal solutions [13], [14]. Employing the crossovers between genetic algorithms and black hole optimization into a hybrid algorithm is seen as an effective method for generating a test case prioritization list, along with the exploitation and exploration capabilities of genetic algorithms and black hole optimization [15], [16], [17]. The subject of a new hybrid genetic-black hole algorithm paper will be closely presented, with the focus on the combination of the GA and black hole optimization powers to stand out with the prioritization outcomes.

The black hole optimization algorithm approach combined with the genetic algorithm (GA) technique put forward here, tries to solve the problem at hand. Genetic algorithms are known traits in the sphere of optimization, while the BHO wielded power, and immersed its users by the social bonds of the humpback whales, has had its presence shown successfully the competition side of various optimization problems. Let's notice that the merging of the meta-heuristics methods opens up a promising line for the solution of all the complexities faced while setting up the test case priorities and optimization. This paper aims to add to existing literature by zoning in on a hybrid approach that takes the both process of genetic algorithms and black hole optimization and further proposes to do this for the smooth prioritization of test cases. The key facets of the paper's contribution can be summarized as follows:

1. The paper discourse on the new way the rotation of black hole and genetics optimization is presented, this work involves the implementation of black hole optimisation and genetics algorithms. This multistage technique considers inefficient test cases generation through the use of genetic algorithms, that is, in the second phase proceeds with a black hole optimization approach aimed to ensure testing of crucial cases selection.
2. The paper focuses on the dynamics of mixed developmental process incorporation on a software project by using the prototypical project study. As the integrated black hole algorithm can be incorporated both in the cross-over and the mutation process and is obviously found to have clearer better performance and as a result solved the problem of greater complexity that human could not spot easily and could be handled by human at best fully.
3. The paper strong points on black hole algorithm integration along with crossover and mutation steps leads to a more probability of successful application as far as contemporary systems are concerned setting aside its pragmatic capability.

In closing, the intelligent prioritization of test cases in software systems is a new breakthrough in the area of testing, which may influence the quality of software testing and the redistribution of resources in large and dynamic software systems. The hybridization of genetic algorithms and the black hole optimization gives raise to interesting opportunities for finding a suitable problem-solving algorithm with respect to difficulties of test case prioritization and optimization.

2. Related Work

In the last couple of years, the fast growth in the size and complexity of software systems has dramatically become a nightmare to manually generate test input, which is quite costly now [18]. Therefore, the researchers investigated the possibility of automatic test data generation through the exploration of the inadequacy of exhaustive enumeration to work for larger programs which are due to the fact the execution of random methods normally ignores important software features that are not covered in the tests [19]. Main challenges in testing data generation are the epitome of unpredictability incurred by huge and complex software systems [20]. To these many challenges, different methodologies have come up such as test case generation from models [21] and utilization of genetic algorithms (GA) [22], [23], simulated annealing (SA) [24] and ant colony optimization

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

(ACO)[25]. Another instance in that context is the work of Ramírez, Romero, and Ventura [26] who designed a search-based algorithm based on Simulated Annealing (SA) which has the possibility to solve complicated problems. Although SA has been in use since metallurgical annealing it has become a useful technique also in computation. Such an approach confirms that the proposed SA is highly malleable, can be effectively applied for multi-objective tasks and extracting system source code design abstractions is an example. They also studied simulated annealing's potential in software architecture design, using a Hybrid Harmony Search and Particle Swarm Optimization Algorithm (HSPSO) for package modularizing and constructing algorithms to auto clean-up which minimizes coupling among packages and cycle dependencies in object-oriented systems. Expressly, software modularization [27] is another technical field on which hill climbing algorithms (HC) are utilized as one of their basic approaches. Even though Genetic Algorithms (GA) are quite powerful in producing test data for path coverage they can encounter challenges, particularly on many targets routes, where [28] was indicated. In addition, the capabilities of the Firefly algorithm (FA) to efficiently maximize the values of mathematical formulas are not comparable to the abilities of other algorithms to recognize code and function statements with high coverage value [29]. Furthermore, the incorporation of Particle Swarm Technique and Genetic Algorithm (GA) [30], and the parallel use of Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) algorithms [31], [32] lead into new, unreliable results in ensuring software optimization, but the concern about convergence time and their non-functional testing remains unexplored.

3. Test Case Prioritization

In software testing, the test-case prioritization is a vital activity, and it is aimed to systematically ordering test case execution in the best dynamics that make the testing process more productive [3]. It implies the systematic generation of test cases under diverse conditions, subject to different success criteria, that helps to point to the tests which probably may have revealed the key flaws the earliest and guide the allocation of the testing resources effectively [33]. This procedural step is the basis because of enumerating the test cases orders as an answer to the possible exhaustive limitations.

Extensive testing, which means that there are all test cases executed without precedence, it is time for the software testers to know that there is a problem waiting for their solution [34]. The test coverage for complex and large scale of software systems is often difficult to achieve because the number of potential test scenarios grows exponentially as systems become more complex. Freely speech about the fact that many of the test cases cannot be seamlessly executed when they belong to large test suites, usually due to the amount of time and resources required, is obvious. In addition to all these, the result is even extended to the detection of faults which occurs in a later form as shown in Figure (1), resulting into high costs and defect in quality of the software [35].

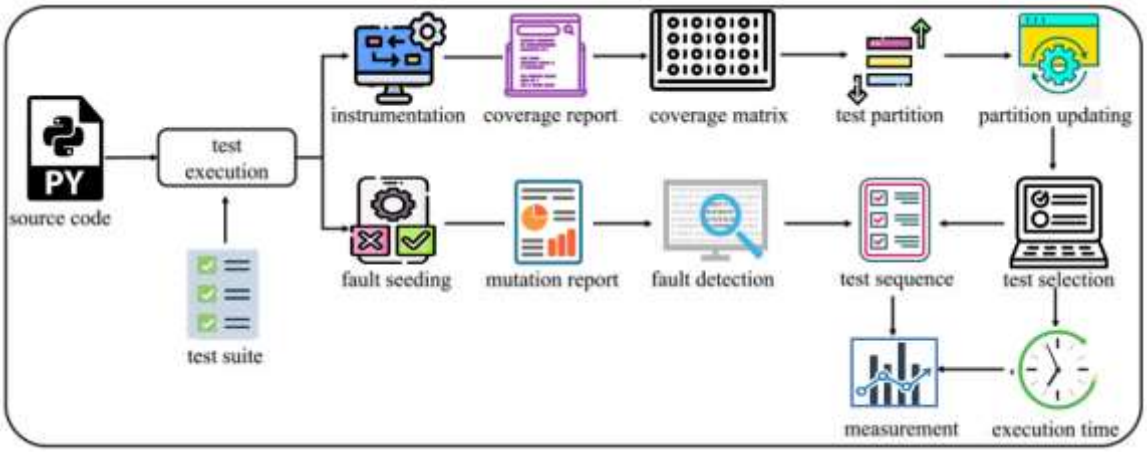

Figure 1: Test Case Prioritization Process.

Test case prioritization is identified as a decisive strategy among the many testing techniques to address the problems raised by a large number of test cases. The testing strategy achievement is implemented by structurally placing the cases of the test in a manner with the purpose defined and by extracting key priorities. It is through this that critical defects are detected and eliminated early [36]. These help to guide the ordering of the testing draws in such a manner that a clear picture of the disease profile of the community is represented. Optimizing the utilization of testing resources will thus be efficient. Prioritizing test cases in essence creates a strategic tool that when applied appropriately makes testing process more efficient by assigning resources effectively and faster detection of the major issues that software tends to have [37], [38].

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

4. Methodology

The proposed hybrid algorithm of genetic black hole (HGBH) for test case prioritization begins by reading test suite data and configuring major parameters including of population size, number of generations, crossover rate, and mutation rate. Having accomplished that, the objective function is stated to become a measuring instrument of test case prioritization solutions fitness. The algorithm opens with the population initialization for the genetic algorithm: random passing of the test cases is produced and for each chromosome they are evaluated by the objective function; the initial generation count is set to 1 as illustrated in Algorithm (1).

Algorithm 1 Hybrid Genetic Black Hole (HGBH)

```
Initialize population P with random solutions (test case orderings)
Evaluate fitness of each solution in P using objective function
Set maximum generations G, population size N
Set genetic algorithm parameters (crossover rate, mutation rate, selection mechanism, etc.)
Set black hole algorithm parameters (number of black holes, migration rate, etc.)

for generation = 1 to G do:
   // Perform genetic algorithm operations
   Select parents from population P based on fitness
   Perform crossover and mutation to generate offspring with black hole optimization:
      // Crossover with Black Hole Optimization
      for each pair of parents do:
         Apply crossover operator to generate offspring
         Evaluate fitness of offspring
         Apply black hole optimization to improve offspring positions:
            for each offspring do:
               Move offspring towards better solutions using gravitational force:
               offspring_position = current_position - (gravitational_force / distance^2)
      // Mutation with Black Hole Optimization
      for each offspring do:
         Apply mutation operator to modify components in the offspring
         Evaluate fitness of the mutated offspring
         Apply black hole optimization to improve the offspring's position:
            Move offspring towards better solutions using gravitational force:
            offspring_position = current_position - (gravitational_force / distance^2)

   Select survivors for the next generation based on combined fitness
end for

Select best solution from final population as the prioritized test suite
```

The algorithm begins with the step of forming a population ***P***, whose every element being a set of random solutions is a different arrangement of test cases. Thus, this step ensures diversity, belonging to those most helpful and appealing to diverse approaches. With regards to evaluation; when the initial population is made, the fitness of each solution is ascertained based on objective function. The tasking of this function is to decide how good are the cases of testing to be put in the sequence based on different factors, either code coverage, fault detection rate or execution time. Fitness scores correspondingly increase as higher genetic information result in the more proficient individuals.

It is necessary to fix both the GA parameters and the BHO the algorithm. These parameter are sample of cross over rate, mutation rate, selection mode, number black holes, migration rate etc. Correct set of these parameters is vital for program work. The algorithm proceeds with genetic algorithm operations for each generation:- The algorithm proceeds with genetic algorithm operations for each generation: They mate with each other giving rise to a new generation with improved fit. This aspect will give more possibilities for higher-fitness solutions to be chosen as parents and therefore increase the probability of successful solutions. Various crossbreeding and mutation operations are introduced into the selected pair to produce new offspring. Crossover can, in this case, be defined as the process of exchanging genetic information from two parents, whereas mutation is associated with random implementation of changes in individual offspring.

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

To observe that, the two tasks are illustrated by black hole optimization enhancement of crossover both steps and mutation. The integration of elder generations of the new breed to the offspring provides enhanced performance plus enduring diversity. During crossover, where the parents pairs get mixed up to form the offspring, each parent submits one copy of their chromosomes to another. At the end, the black hole optimization is run on those children to make sure they are maximally optimized. This method operates in the same manner by moving newborns closer to the optimal solutions based on gravitational forces, which are then used to develop complex functions.

In the same way, in the mutation-step each offspring is altered, and then black hole optimization method applies for correcting the offspring positions by means of the gravitational force calculation. This process reoccurs permanently and is thus leading to a better generation that contributes to well-being. Finally, after a population has a new breed and those individuals have been optimized; survivors of the next generation are ranked based on the integrated fitness from both genetic algorithm and the enhancements from black hole optimization. It makes sure that the population becomes this way as it gets better solutions with time.

The process moves on through generations as defined in the algorithm, till the number of generations has been occupied. Furthermore, the final results from the population will be selected only in light of containing the best case scenario. It defines the perfect sequence of the test cases applying together several black holes and genetic method to meet better results. Genetic algorithms and black hole optimization together breed a successful diversity of techniques that constitute a balance between exploration and exploitation resulting in an efficient priority system for test cases. The blend of these two algorithms brings the efficiency of the algorithms into play so that they can do a thorough search job, yielding the output of high-quality solutions and eventually improve software testing in terms of effectiveness.

Thus, the best solution from the respective final population is chosen and is handed over to the algorithm as the answer in form of the output as shown in Figure (2). A self-learning mechanism is introduced within a genetic algorithm whereby it has an ability to produce highly effective exploratory test cases through an integration of black hole optimization approach.

5. Result and Discussion

The evaluation of performance and the discussion of the results are the main tasks of the procedure. The aim of the analysis is to determine the effectiveness and efficiency of the hybrid genetic-black hole technique for the test case selection. The results of different runs (Mutation, Crossover, Generation) and best solutions obtained as well as the reported analysis are used to conduct the evaluation. Evaluation is centered around a set of important metrics: namely, execution time, quality of the prioritized test suites, Average Percentage Faults Detected (APFD) as shown in Figure (3), and the quality of the test suite.

Running time of an algorithm denotes its computational expenditure for finding the prioritized test suites although it’s an important measure. It is important to notice that the execution timelines differ in the experiment results depending on the run, and the time ranges from about 4.3 seconds to 30.3 seconds as illustrated in Figure (4). This variation could be associated with the factors of the size of test suite, the level of complexity in the definition of the test cases, and environment parameters applied during different runs. Generally, the algorithm works well relative to other algorithm in the sense that the running time of the algorithm is within the reasonable range and also practical for actual application.

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

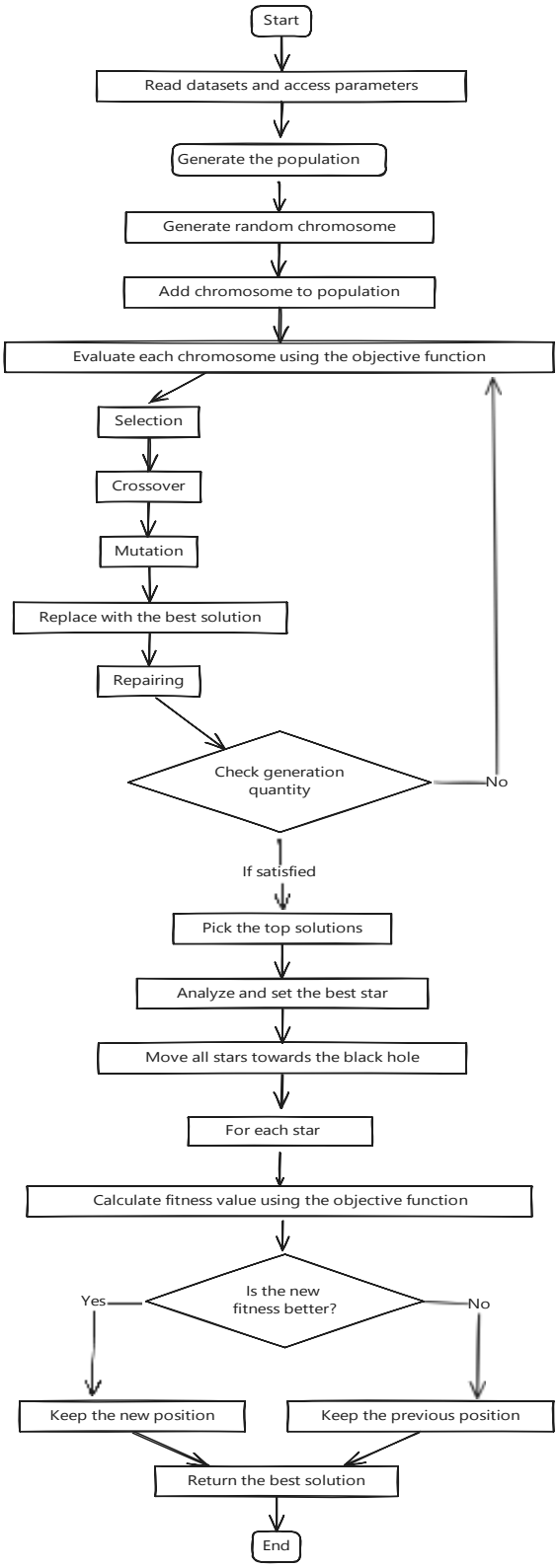

Figure 2: The Proposed Hybrid Genetic Black Hole (HGBH) Algorithm.

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

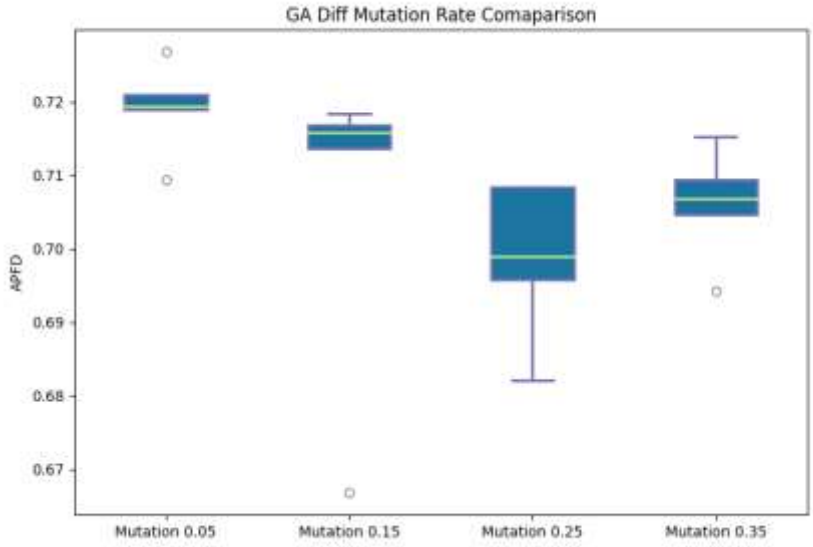

Figure 3: Average Percentage Faults Detected for Different Mutations.

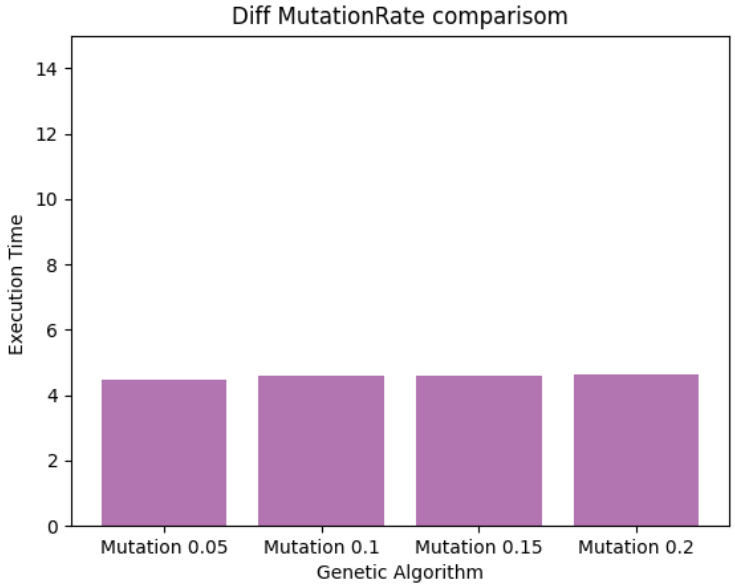

Figure 4: Execution Time for Different Mutations.

Then, follow-up is the fitness function, evaluated as the Average Percentage of Faults Detected (APFD), through which you can understand how good the selection algorithm the prioritized test suits. The most notably APFD scores found in given results are approximately 0.932, 0.919, 0.925 and 0.930 for Crossover 0.6, Crossover 0.7, Crossover 0.8, and Crossover 0.9, correspondingly as shown in Figure(5).

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

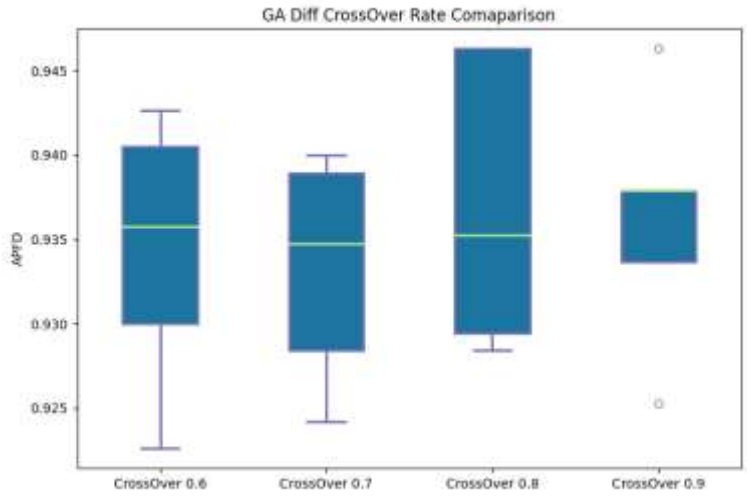

Figure 5: Average Percentage Faults Detected for Different Crossovers.

The values mean that the maximum coverage and the priority putting order of the tests manage to detect a high percentage of faults and find most of the bugs in the software under test. Nevertheless, need to note that the issue arises concerning time consuming verses exactness since it may these calculations to last longer as they nearly equals as described in Figure (6).

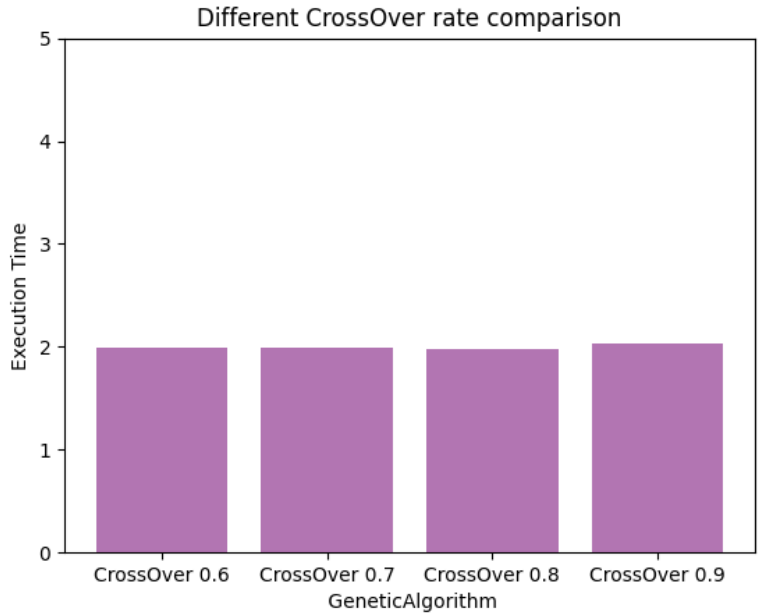

Figure 6: Execution Time for Different Crossovers.

The effectiveness of the ordered test suites generated by the algorithm is also verified by looking at the results of the alternative generations as illustrated in Figure (7). The fittest solutions be those that get detected mostly faults, and stand as ones of high fitness values based on the optimization of test cases to maximize fault detection. Alongside, the particular array of the solutions obtained from the algorithm is an indication of variation in fitness values among the runs and the best solutions explored. If the algorithm fails to consider the diversity of regions of the search space, there is a risk of it converging to suboptimal solutions; thus adequate diversity is equally important.

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

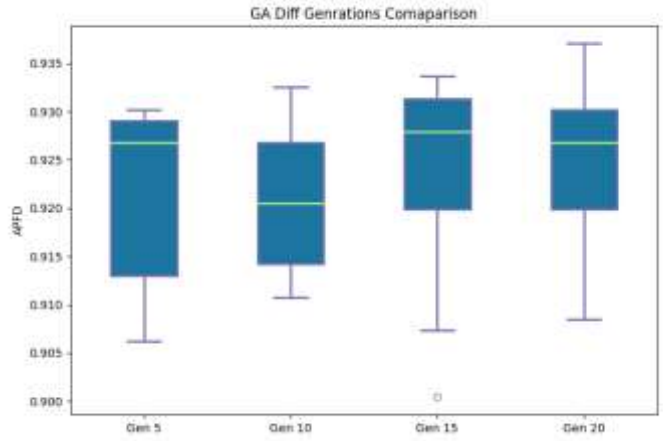

Figure 7: Average Percentage Faults Detected for Different Generations.

On the other hand, by the test statistics, the hybrid algorithm is shown to have decent time performance in test case prioritization and being able to perform well in both decent time performance in the fault detection department and computational efficiency as shown in Figure (8). Another step of further study, proving the algorithm on larger and more representative testing suites, may be the next stage to check its robustness and scalability for any software system and testing tests. Also, a comparative analysis carried out with test cases of the already-existing test case prioritization techniques could provide clarity of the relative benefits and drawbacks of the presented algorithm. In sum, the finding of the hybrid genetic-black hole method points to a promising way for automated testing software process optimization and maintenance quality improvement.

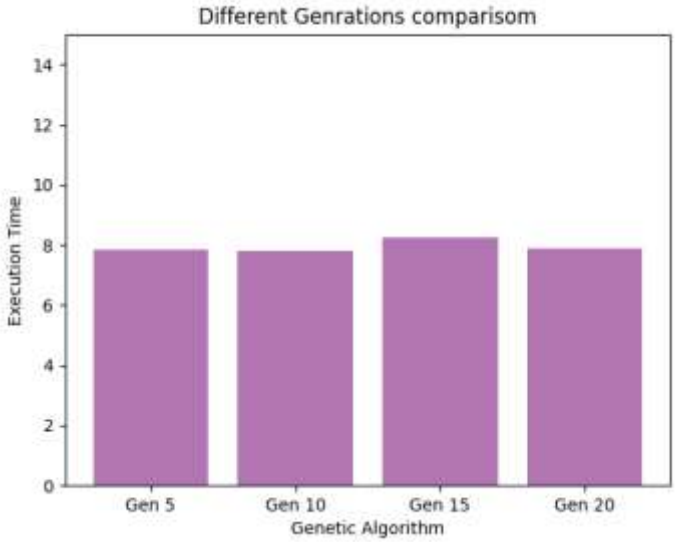

Figure 8: Execution Time for Different Generations.

6. Conclusion

This paper presents a hybrid meta-heuristic technique offering genetic algorithms and black holes optimum approaches in order to allow us to do efficient allocation of resources and find defects early on during the software construction process. The consequence of surpassing the traditional approaches in powering industrial software based on our technique is higher code coverage, an earlier detection of defects, and a resource utilization that achieved maximum efficiency. Though the aim is to show how the proposed approach performed, the results obtained in the APFD value estimations for first item of the selection set with fittest APFD 0.9321 achieved in 6 generations and the mean APFD estimations ranging from 0.9247 to 0.9302 across various procedural datasets showing robustness and effectiveness. Lastly, the direction of test case prioritization in upcoming years might lie in more and more merging of machine learning technologies, coming up with dynamic

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

prioritization methods, and discovering cross domain utilitarian aspects in the software tests to overcome new issues with testing and produce exceptional quality of software systems. Through these innovations not only it could become the one bringing major breakthroughs to the software testing and the optimization, but with it, the proposed approach could greatly affect the whole discipline.

Funding: "This research received no external funding"

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] H. M. Fadhil, M. N. Abdullah, and M. I. Younis, "Combinatorial Testing Approaches: A Systematic Review," Iraqi Journal of Computers, Communications, Control, and Systems Engineering (IJCCCE), vol. 24, no. Accepted, 2023.

[2] K. Sugali, C. Sprunger, and V. N Inukollu, "Software Testing: Issues and Challenges of Artificial Intelligence & Machine Learning," International Journal of Artificial Intelligence & Applications, vol. 12, no. 1, pp. 101–112, 2021, doi: 10.5121/ijaia.2021.12107.

[3] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," Information and Software Technology, vol. 93. Elsevier B.V., pp. 74–93, Jan. 01, 2018. doi: 10.1016/j.infsof.2017.08.014.

[4] K. El-Fakih, A. Alzaatreh, and U. C. Türker, "Assessing test suites of extended finite state machines against model- and code-based faults," in Software Testing Verification and Reliability, 2021. doi: 10.1002/stvr.1789.

[5] O. Achimugu, P. Achimugu, C. Nwufoh, S. Husssein, R. Kolapo, and T. Olufemi, "An Improved Approach for Generating Test Cases during Model-Based Testing Using Tree Traversal Algorithm," Journal of Software Engineering and Applications, vol. 14, no. 06, 2021, doi: 10.4236/jsea.2021.146015.

[6] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "TECHNICAL REPORT : TR-Precrime-2020-03 Testing Machine Learning based Systems : A Systematic Mapping," Empir Softw Eng, 2020, [Online]. Available: http://link.springer.com/10.1007/s10664-020-09881-0%0Ahttps://doi.org/10.1007/s10664-020-09881-0

[7] C. P. Yang, G. Dhadyalla, J. Marco, and P. Jennings, "The effect of time-between-events for sequence interaction testing of a real-time system," Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018, pp. 332–340, 2018, doi: 10.1109/ICSTW.2018.00069.

[8] M. I. Younis, "DEO: A dynamic event order strategy for T-way sequence covering array test data generation," Baghdad Science Journal, vol. 17, no. 2, pp. 575–582, 2020, doi: 10.21123/bsj.2020.17.2.0575.

[9] M. A. S. Mohd Shahrom, N. Zainal, M. F. Mohamad, and S. A. Mostafa, "A Review of Glowworm Swarm Optimization Meta-Heuristic Swarm Intelligence and its Fusion in Various Applications," Fusion: Practice and Applications, vol. 13, no. 1, 2023, doi: 10.54216/FPA.130107.

[10] O. S. Ahmed, F. Fadhil, L. H. J. Alzubaidi, and R. Al-Obaidi, "Fusion Processing Techniques and Bio-inspired Algorithm for E-Communication and Knowledge Transfer," Fusion: Practice and Applications, vol. 10, no. 1, 2023, doi: 10.54216/FPA.100109.

[11] S. Sabharwal and M. Aggarwal, "Test set generation for pairwise testing using genetic algorithms," Journal of Information Processing Systems, vol. 13, no. 5, 2017, doi: 10.3745/JIPS.04.0019.

[12] C. P. Vudatha, S. Nalliboena, S. K. R. Jammalamadaka, B. K. K. Duvvuri, and L. S. S. Reddy, "Automated generation of test cases from output domain of an embedded system using Genetic algorithms," in ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology, 2011. doi: 10.1109/ICECTECH.2011.5941989.

[13] H. N. Nsaif and D. Norhayati Abang Jawawi, "Binary Black Hole-Based Optimization for T-Way Testing," IOP Conf Ser Mater Sci Eng, vol. 864, no. 1, 2020, doi: 10.1088/1757-899X/864/1/012073.

[14] H. N. Nsaif Al-Sammarraie and D. N. A. Jawawi, "Multiple Black Hole Inspired Meta-Heuristic Searching Optimization for Combinatorial Testing," IEEE Access, vol. 8, pp. 33406–33418, 2020, doi: 10.1109/ACCESS.2020.2973696.

[15] T. O. Ting, X. S. Yang, S. Cheng, and K. Huang, "Hybrid metaheuristic algorithms: Past, present, and future," in Studies in Computational Intelligence, vol. 585, Springer Verlag, 2015, pp. 71–83. doi: 10.1007/978-3-319-13826-8_4.

[16] A. P. Agrawal, A. Choudhary, and A. Kaur, "An effective regression test case selection using hybrid whale optimization algorithm," International Journal of Distributed Systems and Technologies, vol. 11, no. 1, pp. 53–67, Jan. 2020, doi: 10.4018/IJDST.2020010105.

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024

[17] H. M. Fadhil, M. N. Abdullah, and M. I. Younis, "TWGH: A Tripartite Whale–Gray Wolf–Harmony Algorithm to Minimize Combinatorial Test Suite Problem," Electronics (Basel), vol. 11, no. 18, 2022.
[18] B. Xu, X. Xie, L. Shi, and C. Nie, "Application of genetic algorithms in software testing," in Advances in Machine Learning Applications in Software Engineering, 2006. doi: 10.4018/978-1-59140-941-1.ch012.
[19] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," in Proceedings - 6th International Conference on Information and Communication Technology for the Muslim World, ICT4M 2016, 2017. doi: 10.1109/ICT4M.2016.40.
[20] S. Varshney and M. Mehrotra, "Search based software test data generation for structural testing," ACM SIGSOFT Software Engineering Notes, vol. 38, no. 4, 2013, doi: 10.1145/2492248.2492277.
[21] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," Software Testing Verification and Reliability, vol. 22, no. 5, 2012, doi: 10.1002/stvr.456.
[22] A. Alert and L. Grunske, "Test data generation with a Kalman filter-based adaptive genetic algorithm," Journal of Systems and Software, vol. 103, no. C, 2015, doi: 10.1016/jjss.2014.11.035.
[23] G. Fraser and A. Arcuri, "1600 faults in 100 projects: automatically finding faults while achieving high coverage with EvoSuite," Empir Softw Eng, vol. 20, no. 3, 2015, doi: 10.1007/s10664-013-9288-2.
[24] R. Matinnejad, S. Nejati, L. C. Briand, and T. Bruckmann, "Effective test suites for mixed discrete-continuous stateflow controllers," in 2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings, 2015. doi: 10.1145/2786805.2786818.
[25] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World, 2014. doi: 10.1109/NOMS.2014.6838365.
[26] A. Ramírez, J. R. Romero, and S. Ventura, "An approach for the evolutionary discovery of software architectures," Inf Sci (N Y), vol. 305, 2015, doi: 10.1016/j.ins.2015.01.017.
[27] Amarjeet and J. K. Chhabra, "Harmony search based remodularization for object-oriented software systems," Comput Lang Syst Struct, vol. 47, 2017, doi: 10.1016/j.cl.2016.09.003.
[28] X. Yao and D. Gong, "Genetic algorithm-based test data generation for multiple paths via individual sharing," Comput Intell Neurosci, vol. 2014, 2014, doi: 10.1155/2014/591294.
[29] R. K. Sahoo, D. P. Mohapatra, and M. R. Patra, "A Firefly Algorithm Based Approach for Automated Generation and Optimization of Test Cases," International Journal of Computer Sciences and Engineering, vol. 4, no. 8, 2016.
[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 (IJIET), vol. 3, 2014.
[31] O. R. Olaniran and M. A. A. Bin Abdullah, "Bayesian variable selection for multiclass classification using bootstrap prior technique," Austrian Journal of Statistics, vol. 48, no. 2, 2019, doi: 10.17713/ajs.v48i2.806.
[32] O. R. Olaniran and W. B. Yahya, "Bayesian hypothesis testing of two normal samples using bootstrap prior technique," Journal of Modern Applied Statistical Methods, vol. 16, no. 2, 2017, doi: 10.22237/jmasm/1509496440.
[33] A. Singh, R. K. Bhatia, and A. Singhrova, "Machine learning based test case prioritization in object oriented testing," International Journal of Recent Technology and Engineering, vol. 8, no. 3, 2019, doi: 10.35940/ijrte.C3968.098319.
[34] M. M. Sharma, A. Agrawal, and B. Suresh Kumar, "Test case design and test case prioritization using machine learning," Int J Eng Adv Technol, vol. 9, no. 1, pp. 2742–2748, Oct. 2019, doi: 10.35940/ijeat.A9762.109119.
[35] M. Khatibsyarbini et al., "Trend Application of Machine Learning in Test Case Prioritization: A Review on Techniques," IEEE Access, vol. 9, pp. 166262–166282, 2021, doi: 10.1109/ACCESS.2021.3135508.
[36] R. Lachmann, M. Nieke, C. Seidl, I. Schaefer, and S. Schulze, "System-level test case prioritization using machine learning," in Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016, Institute of Electrical and Electronics Engineers Inc., Jan. 2017, pp. 361–368. doi: 10.1109/ICMLA.2016.163.
[37] A. Bajaj and O. P. Sangwan, "Test Case Prioritization Using Bat Algorithm," Recent Advances in Computer Science and Communications, vol. 14, no. 2, pp. 593–598, Mar. 2019, doi: 10.2174/2213275912666190226154344.
[38] M. Mann, P. Tomar, and O. P. Sangwan, "Test Case Prioritization Using Metaheuristic Search Techniques," Arab Gulf Journal of Scientific Research, pp. 157–173, Dec. 2015, doi: 10.51758/agjsr-04-2015-0015

Doi: https://doi.org/10.54216/FPA.140221
Received: July 13, 2023 Revised: December 18, 2023 Accepted: February 07, 2024