International Journal of Engineering & Technology, 7 (4.28) (2018) 368 -391

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Effective Search-Based Approach for Testing Non-Functional Properties in Software System: an Empirical Review

N.M. Bala[1], S.Suhailan[2*]

Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin Kuala Terengganu, Terengganu, Malaysia
**Corresponding author E-mail: suhailan@unisza.edu.my*

Abstract

Search-based software testing (SBST) is considered an effective process in the generation of non-functional test cases. The SBST employs metaheuristic search techniques to evaluate the best-case and worst-case execution times of real-time scenarios. However, these search techniques suffer software re-modularization for software systems. In this paper, an exploratory review on some of these techniques such as Genetic Algorithms, Harmony Search and Simulated Annealing is presented by highlighting the fitness function employed, non – functional testing and the challenges observed. The review also investigates each technique based on different applications employed in a white box, black-box, and gray-box testing. It shows that Harmony Search-Based Algorithm is the effective technique to address the problem of software re-modularization.

Keywords: *Genetic Algorithms; Harmony Search; Non-Functional Testing and Simulated Annealing.*

1. Introduction

Software testing is the most important segment of SDLC to determine if applications are performing like they are intended to, ensure high quality and reliability of the software. Testing is the most widely used validation approach, but it is largely ad hoc, expensive and most effective (1). Two of the major categories of tests are functional and non-functional testing. Non-functional (NF) tests are used to determine the fitness of the application for example how fast a product responds to a request or how long it takes to do an action (2). Metaheuristic search is among the techniques used in non-functional testing especially in generating test cases. A widely applicable and effective way of generating test data and optimizing the testing process is implemented using a Search-based Software Test (SBST). This review allows us to identify the extent to which the non-functional properties are applicable and provides an overview of existing non-functional properties tested using metaheuristic search techniques. The objective of this review is, therefore, an exploration and analogy of non-functional properties tested using metaheuristic search techniques and identification of effective technique used to test individual non-functional property.

2. Search Base Software Testing

SBST has been applied to a variety of test case design methods as follows, white box, black box and grey box testing. The objective of undertaking this review is to examine existing work into SBST.

2.1 Metaheuristic Search Technique

Metaheuristic search techniques are widely used in diverse applications of scientific domains among which software testing is a significant application. This has been used in distinct stages that begin with planning to execution. This paper will investigate different applications of metaheuristics employed in a white box, black-box and grey-box testing. Also, metaheuristic search techniques are considered to play a significant role in the domain of non-functional testing in evaluating the best-case and worst-case execution times (BCET, WCET) of real-time scenarios.

We are targetting the types of non-functional testing that use Metaheuristic search techniques such as Genetic Algorithms, Harmony Search and Simulated Annealing (2), different fitness functions are used in different types of search-based non-functional testing including the challenges in the application of these techniques.

In Chen (3) discussed that Genetic Algorithms begins with a set of the first generation, which are sampled at random from the problem domain. The algorithms are developed to perform a series of operations that transform the present generation into a new, fitter generation. In each generation was evaluated with a fitness function to approach the best solution. Genetic algorithms operations are designed to produce an efficient solution for the target problem.

In Wegener et al.(4), genetic algorithms were used to search for input situations that produce very long or very short execution times. The experimental results using a simple C function showed that the longest execution time of 26.27 µs was found very quickly with GA in less than 20 generations and a new shortest execution time of 5.27 µs, which was not discovered by statistical and systematic testing, was found.

In Alander et al.(5) experiments were performed in a simulator environment to measure response time extremes of protection relay software using genetic algorithms. The results showed that GA generated more input cases with longer response times.

In Briand et al.(6), genetic algorithms were used for critical deadlines misses. The author defined this as stress testing also known as robustness testing. The objective was to find the sequence of

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

arrival times of events for aperiodic tasks, which causes a delay in the execution of the target task.

In Mutoh et al.(7), GA is applied to various real-world applications, the search space is unknown, and many applications need large amounts of execution time. The increase in some crossovers makes it easy to find high-fitness individuals because of the spread of the search space. However, the increase of the prediction error may cause the probability of selecting precisely best two individuals per crossover to decrease.

Thus, it is important to increase the predictive accuracy to improve the search performance by increasing the number of crossovers. However, it requires improvement in the effectiveness of the proposed method.

In Tracey & Clark (8), applied simulated annealing (SA) to test four simple programs for worst-case execution times (WCET) as part of a generalized test data generation framework. The fitness function used is the measure of actual dynamic execution time. The WCET of the programs was already known, and a valid test case was one that exercised a path yielding the already known WCET. The experimental findings show that the use of SA was more effective with a larger parameter space.

In Tracey & Clark (8), applied simulated annealing (SA) to test four simple programs for worst-case execution times (WCET) as part of a generalized test data generation framework. The fitness function used is the measure of actual dynamic execution time. The WCET of the programs was already known, and a valid test case was one that exercised a path yielding the already known WCET. The experimental findings show that the use of SA was more effective with a larger parameter space.

The authors highlighted the need for a detailed comparison of various optimization techniques to explore WCET and best case execution time (BCET) of the software under test. The highlight of the paper is to show the usefulness of the amount of search space reduced.

In Lu et al.(9) proposed a method which is base on the simulated annealing algorithm for the web-based testing environment to evaluate the performance of web-based testing, this study conducted a series of experiments. On the efficiency of items selection, the execution time to SA search, exhaustive search, and random search method was compared to implement the SA algorithm assessment. The below figure shows the average execution time of the SA search, exhaustive search, and random search, and SA search uses different iterations. From the figure, we can see that item selection at the time of each method is constantly increasing with the increase of the scale of the item pool, but the degree of growth is different.

In Arockiam (10), SA was used for the step-wise construction of test scenarios that test safety properties in cyclic real-time control systems. The control system requires sequential behaviour of the step-wise construction as it was expected that safety property violation would occur after execution of a particular trajectory or sequence of data in the input domain. The solution space was divided into several subsets of smaller sizes where different classes of test sequences were independently searched. For each class, the objective was defined into sub-objectives corresponding to either safety property violation or the achievement of a dangerous situation.

In Ramírez et al.(11), develop a search-based algorithm simulated annealing (SA) such that utilized to solve complex optimization problems. This search-based meta-heuristic technique was employed for the annealing process in metallurgy. The SA algorithm was further used to extract the design abstractions from source code and demonstrated the usability of SA algorithm in multiple problem instances. The proposed algorithms were compared with other search-based algorithms and apply the SA algorithm for automated software refactoring to make the software more maintainable.

In Nur & Radzi (12) a hybrid Back Propagation and Harmony Search Algorithm called BPHSAMLP is employed for Neural Networks(NNs) of the feed-forward type to classify problems. Five classification benchmark datasets were used to test and verify the training performance and generalization ability of the BPHSA scheme. The sum of squared errors, training time and accuracy were compared with standard HSA and standard BP. The results show that BPHSAMLP scheme can successfully train feed-forward type NNs with reasonable time, MSE and high accuracy. The BPHSA-MLP is better than compared algorithms in training and classification of test patterns. Therefore, NN type of feed-forward classification problems can be trained with BPHSA technique. The scheme can train both supervised and unsupervised models.

In Qin et al.(13) defined the opportunities of simulated annealing in designing software architecture and further applied the algorithms to modularize the source code classes into packages. This approach was developed by automatically minimizing the package coupling and cycle dependencies of an object-oriented software system.

The total cost was the minimum value of the different sub-objective cost functions. The efficiency of using SA was analyzed in comparison with random sampling. The results confirmed the usefulness of stepwise construction of test scenarios, but regarding the efficiency of the SA algorithm, the cost-effectiveness as compared with random sampling remains questionable. Recently a metaheuristic algorithm defined by Harmony Search (HS) is found to be widely used as an effective and convenient approach to solving numerous issues related to diverse domains of science and engineering problems.

However, the application and studies based on the HS algorithm have not been conducted by researchers while considering non-functional testing. A recent study indicates the employment of metaheuristic search technique to generate t-way test suite is Harmony Search Strategy (HSS). Subsequently, the research ascertained that in Comparison to the existing approaches such as Particle Swarm Optimization (PSO), only a minimum computation amount is required by the HS algorithm with only a few parameter settings. Further, this approach is found to support uniform and variable interaction strength with high interaction strength integrating around 14 constraints. Besides, the HS algorithm was used to develop an efficient system such as HSS and HS-PTSGT (14) defined as a pairwise generator tool.

A new method artificial intelligence optimization technique (HS) has been proposed to be applied for the tuning of the classical PID order controller in case of centralized and decentralized control schemes. This demonstrates the effectiveness of the optimization technique in tuning this type of controllers for this model and wellness of decentralized control scheme in comparison with the centralized one (15).

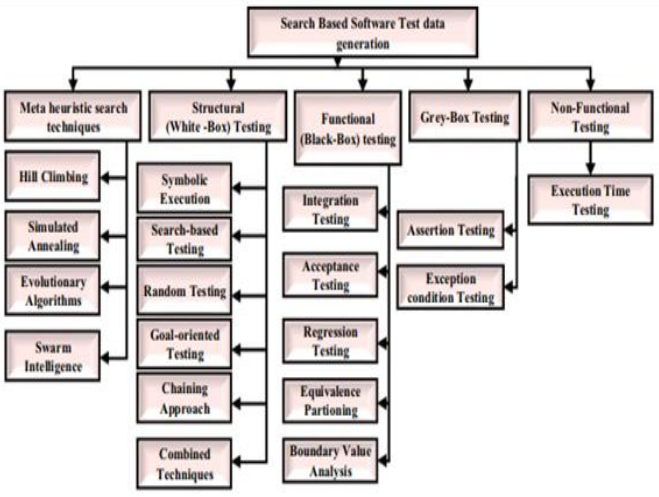


Fig. 1. Testing techniques in the search-based software test data generation scheme

In Sambariya & Shrangi (16), proposed Controller tuned with a harmony search algorithm for controlling the frequency and Tie-

line power responses of a non-re-heat two area power system. This proposed tuned PID is compared with a conventional controller by settling-time, peak over-shoot and peak under-shoot to evaluate the performance.
The recent research proposed a Harmony Search-Based Re-modularization Algorithm (HSBRA) to solve the challenges seen in the non-function testing and software re-modularization for object-oriented software (OOS) systems. These were developed by considering numerous factors related to efficient encoding of the harmony memory, initialization and developing an effective approach for the improvement of a better harmony. These factors were considered in the development of different variants of HSBRA such as HSBRA1, HSBRA2, HSBRA3, and HSBRA4. This research is considered to incorporate fitness function which considers coupling, cohesion, package count index and packages size index. These algorithms were developed based on linear and exponential changes in Harmony Memory Consideration Rate (HMCR) and Pitch Adjusting Rate (PAR) (17).

2.2 Structural (white -box) Testing

The idea of the structural technique in software-based search will be to generate test data for the completion of the execution in the prescribed paths. Structural testing is a key role in the miller and Spooner approach which has been application area that has involved the best attention in search-based software testing. The idea has been undertaken to develop fitness function for execution path, branch coverage and data flow coverage amongst others. Specifically, the program under test which is instrumented and the simulated output through which the categorized structural testing has been deployed and the idea have been in the generation of the paths created for the certain stage of the inputs. The path generated by the evaluation phase of the generation has been instrumented and the simulation completed with inputs which result in the search. The idea has been generated a path and through the program is compared with some better structure of interest for which the cover is sought.
The evaluated function is a specific routine in the idea through the routine for which coverage might be sought using search-based techniques. Chemical solvents have assigned the functionality of this approach. A service to discover every chemical described in the scientific literature and involves checksum estimations. Familiar population generation has been included to improve the research objectives through which the idea have interrogated in the machine concept on which the logic is analyzed. The two execution levels on the work will be based on the continuation of the concept in the logic which has been developed in the simulation results in the research. The equivalent number of the target for the structured input is the number of levels left undisturbed by the sketch made by path completion to the target and the regularly structured simulations. It is equivalent to the number of levels of nesting left unpenetrated by the path en route to the target for structured programs.
From the internal structure of the software under test (18), the white-box testing or structural testing is derived. Through the use of metaheuristic method, specific accomplishments in automating structural test data generation are made. Earlier related approaches are associated with this approach. Before this, reviews in some simple ideas are done. It is arranged in the middle of levels of nesting quit unpenetrated by the path en route to the target for structured programs. Foreigner lay contrivance of the software under test (18), the white-box inquisition or structural assessment is derived. The conformable of metaheuristic attitude specific accomplishments in automating structural test data generation are made. Antique helper approaches are associated with this approach. On this, reviews in some simple ideas are done. Non-representational movement, several of the focal reasons in automatic test input generation is developments in representational execution that has become more relevant.
In its A- regular dawning, instead of concrete inputs the metaphysical consummation executes a program dislike symbolic execution. The station inputs become an absent-minded vehicle, the enactment to carry out rove end is at the end of time literal as a normal of the tram in a conjunctive form called the path condition (PC). At any point in the calculation, the program consists of a symbolic state which expressed as a function of the inputs.
Search-based scrutiny, In any event, the symbolic implementation methods stuffy the smart bulk of indications in our colleagues' responses, where test input generation methods are more commonly search-based software testing (SBST), the second significant number of indications went to research on search-based test input generation practices. By using SBST methods, Harman and colleagues produce the win out over recent in a line of reviews which is concentrating on utilization in software engineering for all purposes.
Diverse adjustment reviews are barring available, including (19), (20 - 24). They also do to the singular regularly in which industrial organizations such as Daimler, Microsoft, Nokia, Ericsson, Motorola and IBM have considered the use of SBST techniques. Because of the extend decade haphazard testing (RT) is another automated test input generation method that has developed significantly. This rise is to superintend the often devastatingly enormous amount of test inputs generated (25), and efficaciousness is attained by defining methods that can either develop the undirected input generation procedure (26). Adaptive random testing is the example of new random testing approaches. Adaptive random testing (ART) (27) is a classification of analyzing methods in which increasing the assortment of the test inputs executed across a program's input domain is used to develop the failure detection efficiency.
Goal-oriented approach: this approach was developed by Ferguson & Korel (28) which ascertained the significance of Implementation of a path. The path is defined to be chosen for individuals for satisfying the physical standards such as statement coverage. These limitations were found to be eliminated by employing a goal-oriented method(29).
Chaining approach: This approach is found to consider an occasion series as an intermediary means for implementing up to the target node to determine the type of trail required (30). This research discusses the process involved in the implementation of a succession of program nodes as an event arrangement. The first set of the sequence is found to consider the begin node and target node. This procedure is found to be efficient in the injection of additional nodes into this event arrangement when difficulties are confronted by the test data search.

2.3 Functional (Black-Box) Testing

Recent research ascertained the most effective example of search-based functional testing as the testing of the car parking controller (31). This controller is used for the identification of a suitable parking space, and for automatically manoeuvring the car without colliding with external objects found in the parking space. In this research, the controller was validated by employing the simulation using Search-Based Software Testing. The fitness function considered the research was mentioned as the shortest distance to the point of collision while parking. This search is utilized to minimize the overall distance to reveal situations where the controller is found to have faults that lead to a collision.
The controller is found to simulate the evolutionary search used generated parking scenarios through the simulation to determine the shortest distance involved in the recorded collision to give a fitness value. The initial version of the system is considered to determine the faults about the position of the car found to be close to the exterior object before the occurrence of the collision. Nikolić (32) indicate the application of metaheuristic search methods for performing logical behaviour analysis of a system. Black-box testing was considered as an appropriate technique for determining this analysis in the absence of data inside components.

Integration testing is defined by the incorporation test were equipment parts, programming segments will be consolidated and assess the interrelation between them. The units are found to co-operate with each other as a more significant code base utilizing both high-contrast box testing strategies. Further, as the devices are found to operate independently does not imply that this operation to be coordinated and collected. Also, interfaces are not found to be regarding the indicated, messages and will not be transferred appropriately. These would lead to the data loss in an interface. These mix test cases will consider analyzers take a gander at low-level and high-level configuration archives (33).

Acceptance testing: Acknowledgment testing is not considered as a framework for fulfilling the acknowledgement criteria where the criteria defined should be acknowledged by a client. Also, formal trying is found to empower the client to determine to lead to figuring out whether to acknowledge or not to the framework. The test group to keep running before endeavouring to convey the item and the client regularly predetermines these tests and given to the test group. If the acknowledgement test cases do not pass, the client maintains whatever authority is needed to decline conveyance of the product. Clients do not indicate a 'complete' arrangement of acknowledgement experiments. The experiments were not a viable replacement for making a particular method of practical framework test cases. The client determines the grandest experiment for every prerequisite. However, a more significant number of tests is required for better understanding. The client acknowledgement test cases are outrun with the objectives to build certainty working in a particular client area at a point of conceivable.

Gray-Box Testing: Gray-box testing is considered as a correlation between practical and necessary data for the motivations behind testing. Gray-box testing (dim box testing) is used to calculate motivations while actualizing those tests at the client or discovery level and incorporates information from inner information structures behind the controlling tests. In this system, it was noticed that the analyzer does not require to utilize the product's source code. The framework under test comprises of data, and yield output that is outer of the 'black-box' which would otherwise, control the information and arrange yield that is not suitable as grey-box. This qualification is vital when directing incorporation testing between two modules of code composed of two unique engineers, where just the interfaces are uncovered for the test. Dim box testing is found to be integrated with parameters such as occurrence, figuring out the limitations of qualities or anomaly messages.

Gray-box testing is utilized to restrict information and to test the application of the inside workings. In software testing, testing an application conveys a great deal of weight and mastering the area of a framework depends on giving the analyzer an edge over the constrained space information. These observed to be dissimilar to grey-box testing. Also, the analyzer has admittance to archives the database and that are found to be dissimilar to discovery testing, where the analyzer tests the application's client interface. In this process, the analyzer will initiate the test information and test situations when making the test arrangement.

Assertion testing: Assertions applied to the different state of a calculation specifies few restrictions. The faults that have been detected in the program is estimated to be false during the declaration. Assertions are entrenched within comment areas similar to Boolean conditions. An above variable statement is used during declarations that are entrenched as blocks of executable code. The incorrect state of the constitution or correct state of the statement is identified while assigning true or false values.

Chaining approach is considered as the process involving the generation of test data. Also, Korel (34) tool automatically creates statements for run-time mistakes as opposed to entrenched programmer assertions such as array boundary violations, division-by-zero and overflow errors. Variables are uninitialized as the machine catch input data for motivating error conditions that are used in the following program statement. In this declaration, nine original Pascal programs are found to be embedded during the initial experiments also an approximate of twenty-five defective versions were found to be manufactured.

Exception condition testing: An exception-related code is found to deviate from the first logic of the program as these languages afford explicit exception-handling concepts. Also, the omission is considered as a means of integrating the run-time faults within the languages such as C++, Java and Ada. The research conducted by Petke et al. (35) found to produce an extensive test data for the essential coverage of the exception handler which is found to increase the omission. As with the effort of Korel (34), both complications were seen to be moderated to the challenges incurred in the sequence of statements through the code or by the execution of a specific statement. The declaration was found to activate the exception through a throw or raise statement. The trials were commenced by considering the seven simple programs comprising of no more than two hundred lines of code. The test data were generated by employing metaheuristic methods for increasing all the exception conditions contained in this code, and further will complete the overall branch coverage of exemption handlers. Also, a diverse variety of exception conditions could be raised by the production of test data as the input situations produced were not during specific operation of the system.

2.4 Non-Functional Testing

Recently, the review conducted on Non-functional testing indicates various applications for test generations such as search-based techniques for white-box testing, black-box testing, grey-box testing (36). These comprehensive reviews indicate that non-functional testing which is based on the search-based techniques will be applicable for the execution of real-time systems.

Quality of service (QoS): A non-functional error will occur if a QoS requirement is violated an example of a non-functional error is the exceeding of the required response time. These are a significant non-functional error regarding real-time systems (37). Furthermore, an exploratory survey of the state-of-the-art in search-based software testing for non-functional system-level properties was discussed in the research conducted by Harman et al.(38) as shown in Fig 2.

In this paper, test sets are generated by occurring faults, and with a genetic algorithm, the tests are applied at randomly chosen points in time and are performed centrally on the test system which operates as a client of the services to be tested. In Canfora et al.(39) genetic algorithms were used to determine the set of bindings between abstract and concrete services that lead to QoS constraint satisfaction while optimizing the objective function. The QoS attributes of composite services were determined using rules with aggregation function for each workflow construct. The fitness function was designed in a way to maximize some QoS attributes (e.g. reliability and availability) while minimizing others (e.g. cost and response time). The findings show that when the number of concrete services is small, integer programming outperformed genetic algorithms. However, as the number of concrete services increased, a genetic algorithm was able to keep its time performance while integer programming grew exponentially.

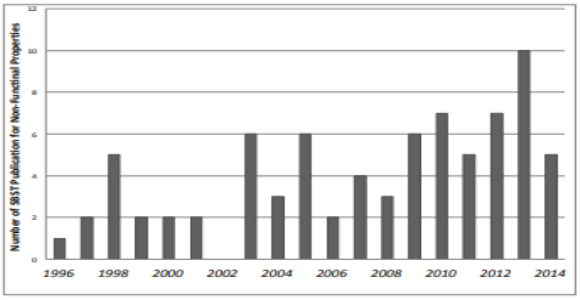


Fig. 2. SBST for non-functional properties were data collected using the SBSE repository (Harman et al.(38)

Security: In Kayacik (40) Grammatical Evolution (GE) was used to discover the characteristics of a successful buffer overflow. The example vulnerable application, in this case, performed a data copy without checking the internal buffer size. The limitation is that the shellcode or the attacker's arbitrary code needs to be modified to increase the success chances of the malicious buffer.

In Del Grosso et al.(41) propose a dynamic weight fitness in which the genetic algorithm weights were calculated by solving a maximization problem via linear programming. The limitation of this paper is a dependency on tools for static analysis and program slicing. Use of instrumentation is a probable obstacle in expanding the approach for larger case studies. The additional computational time required for linear programming calculation.

Execution time testing: The fundamental process of a real-time system is seen to depend not only on the timing behaviour but its reasonable behaviour. The improper timing behaviour of a real-time system is found to occur if outputs are generated too early or too late. Therefore, it is necessary to consider whether it is compliant with its timing limitations by determining the best-case execution time (BCET) and the worst-case execution time (WCET) of a system. Also, the timing behaviour of a piece of software is found to be dependent on its interior arrangement, as well as the features involved in the actual hardware. Nevertheless, this task is viewed to be considered hard regarding implementation. The commands employed at the software stage and their equivalent data items are noticed to be dependent on the time. Also, the compiler could announce the effects at source code level. The movements of the target processor which is enormously complicated are found to be accounted at the hardware level when pipelining and caching processes are deliberately required. The most prolonged or shortest execution times will not yield the longest or shortest paths through the program.

Also, the process of the optimization process is related to transversely different designing and logical censures. Also, search-based software testing is found to be connected from booking to usage. Subsequently, it is necessary to portray great consideration and avoidance principles. Recent studies indicate that software advancement and development, do not report the use of metaheuristic approaches such as tabu search, evolutionary methods, swarm intelligence, hill climbing, simulated annealing and ant colony methods. Besides, it is difficult to identify optimization systems with software testing to portray search-based testing approaches such as white-box (structural), gray-box (combination of functional and structural) where a basic test standard is utilized to test non-functional properties) or black-box (functional).

Therefore, from the exhaustive review, it can be observed that the significance of search-based optimization techniques while it is utilized for testing non-functional properties. Thus, it is necessary to consider the existing approaches to develop an appropriate NFSBST with an accurate, fair and partial manner based on applications.

3. Conclusion

This paper presents a comprehensive study of various effective search techniques for testing the non –functional properties in software testing. The study shows that there are several advantages of employing Software Testing, which eliminates the use of manual testing, reduces efforts and improves reliability. Furthermore, it also demonstrates that the Search Based Software Testing were implemented in various formats such as White Box and Black Box Testing. In addition, these techniques were found to exploit the potential of modern day computing approaches that includes evolutionary algorithms such as Genetic, simulating annealing and harmony search. Finally, the paper suggests that while considering metaheuristic approaches for search-based testing for non-functional properties, the Harmony search algorithm is found to function effectively in terms of reliability and accuracy.

Acknowledgment

This work was supported by Petroleum Technology Development Fund (PTDF) Nigeria.

References

[1] Bertolino A. Software Testing Research : Achievements , Challenges , Dreams Software Testing Research : Achievements , Challenges , Dreams. IEEE Trans Softw Eng. 2007;(September):85–103.

[2] Afzal W, Torkar R, Feldt R. A systematic mapping study on non-functional search-based software testing. Proc Twent Int Conf Softw Eng Knowl Eng. 2008;(2008):488–93.

[3] Chen Y, Zhong Y. Automatic path-oriented test data generation using a multi-population genetic algorithm. In: Proceedings - 4th International Conference on Natural Computation, ICNC 2008. 2008. p. 566–70.

[4] Wegener J, Sthamer H, Jones BF, Eyres DE. Testing real-time systems using genetic algorithms. Softw Qual J. 1997;6(2):127–35.

[5] Alander J, Mantere T, Moghadampour G, Matila J. Searching protection relay response time extremes using genetic algorithm-software quality by optimization. 1997;

[6] Briand LC, Labiche Y, Shousha M. Using genetic algorithms for early schedulability analysis and stress testing in real-time systems. In: Genetic Programming and Evolvable Machines. 2006. p. 145–70.

[7] Mutoh A, Nakamura T, Computation SK-…, CEC' 2003., 2003 undefined. Reducing execution time on genetic algorithm in real-world applications using fitness prediction: Parameter optimization of SRM control. ieeexplore.ieee.org.

[8] Tracey N, Clark J, IFIP KM-P of the, 1998 undefined. The way forward for unifying dynamic test-case generation: The optimisation-based approach. eprints.whiterose.ac.uk.

[9] Lu P, Cong X, Zhou D. The Research on Web-Based Testing Environment Using Simulated Annealing Algorithm. Sci World J. 2014;2014:1–12.

[10] Arockiam L, e-Service NS-IJ of u-and, 2012 undefined. Simulated annealing versus genetic based service selection algorithms. earticle.net.

[11] Ramírez A, Romero JR, Ventura S. An approach for the evolutionary discovery of software architectures. Inf Sci (Ny). 2015;305:234–55.

[12] Nur A, Radzi N, of SS-IJ, 2015 undefined. Near Optimal Convergence of Back-Propagation Method using Harmony Search Algorithm. iaescore.com.

[13] Qin Z, Denker G, Giannelli C, Bellavista P, Venkatasubramanian N. A software defined networking architecture for the internet-of-things. IEEE/IFIP NOMS 2014 - IEEE/IFIP Netw Oper Manag Symp Manag a Softw Defin World. 2014;

[14] Xiang LY, Alsewari AA, Zamli KZ. Pairwise Test Suite Generator Tool Based On Harmony Search Algorithm (HS-PTSGT). UmpirUmpEduMy. 2015;2:62–5.

[15] Omar M, Ebrahim MA, AbdelGhany AM, Bendary F. Tuning of PID controller for load frequency control problem via harmony search algorithm. Indones J Electr Eng Comput Sci. 2016;1(2):255–63.

[16] Sambariya DK, Shrangi S. Optimal design of PID controller for load frequency control using harmony search algorithm. Indones J Electr Eng Comput Sci. 2017;5(1):19–32.

[17] Amarjeet, Chhabra JK. Harmony search based remodularization for object-oriented software systems. Comput Lang Syst Struct. 2017;47:153–69.

[18] Harman M, McMinn P, de Souza JT, Yoo S. Search Based Software Engineering: Techniques, Taxonomy, Tutorial. In Springer, Berlin, Heidelberg; 2012. p. 1–59.

[19] Afzal W, Alone S, Glocksien K, Torkar R. Software test process improvement approaches: A systematic literature review and an industrial case study. Vol. 111, Journal of Systems and Software. 2016. p. 1–33.

[20] Ali S, Briand LC, Hemmati H, Panesar-Walawege RK. A systematic review of the application and empirical investigation of search-based test case generation. Vol. 36, IEEE Transactions on Software Engineering. 2010. p. 742–62.

[21] Ali S, Briand L, … HH-IT, 2010 undefined. A systematic review of the application and empirical investigation of search-based test case generation. ieeexplore.ieee.org.

[22] Arcuri A. It does matter how you normalise the branch distance in search based software testing. In: ICST 2010 - 3rd International Conference on Software Testing, Verification and Validation. 2010. p. 205–14.
[23] Blanco R, Tuya J, Adenso-Díaz B. Automated test data generation using a scatter search approach. Inf Softw Technol. 2009;51(4):708–20.
[24] Harman M. Search Based Software Engineering for Program Comprehension. Progr Comprehension, 2007 ICPC '07 15th IEEE Int Conf. 2007;3–13.
[25] Michael C, McGraw G. Automated software test data generation for complex programs. Proc 13th IEEE Int Conf Autom Softw Eng (Cat No98EX239). 1998;1–12.
[26] Kotelyanskii A, Kapfhammer GM. Parameter tuning for search-based test-data generation revisited: Support for previous results. In: Proceedings - International Conference on Quality Software. 2014. p. 79–84.
[27] Moadab S, Rashidi H. Automatic path-oriented test data generation by boundary hypercuboids. J King Saud Univ - Comput Inf Sci. 2016;28(1):82–97.
[28] Ferguson R, Korel B. The chaining approach for software test data generation. ACM Trans Softw Eng Methodol. 1996;5(1):63–86.
[29] Shahamiri SR, Kadir WMNW, Mohd-Hashim SZ. A comparative study on automated software test oracle methods. In: 4th International Conference on Software Engineering Advances. 2009.
[30] Ferguson R, and BK-AT on SE, 1996 undefined. The chaining approach for software test data generation. dl.acm.org.
[31] Bringmann E, Krämer A. Model-based Testing of Automotive Systems. In: ICST '08 Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation. 2008. p. 485–93.
[32] Nikolić I. How to use metaheuristics for design of symmetric-key primitives. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2017. p. 369–91.
[33] Lefticaru R, Ipate F. A comparative landscape analysis of fitness functions for search-based testing. In: Proceedings of the 2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2008. 2008. p. 201–8.
[34] Korel B. Automated software test data generation. IEEE Trans Softw Eng. 1990;16(8):870–9.
[35] Petke J, Haraldsson S, Harman M, langdon william, White D, Woodward J. Genetic Improvement of Software: a Comprehensive Survey. IEEE Trans Evol Comput. 2017;(c):1–1.
[36] McMinn P. Search-based software test data generation: a survey. Softw testing, Verif Reliab. 2004;14:1–58.
[37] Meixler S, Brinkschulte U. Test Case Generation for Non-functional and Functional Testing of Services. Object/Component/Service-Oriented Real-Time Distrib Comput (ISORC), 2010 13th IEEE Int Symp. 2010;245–9.
[38] Harman M, Jia Y, Zhang Y. Achievements, Open Problems and Challenges for Search Based Software Testing. 8th Int Conf Softw Testing, Verif Valid (ICST), 2015 IEEE. 2015;(Icst):1–12.
[39] Canfora G, Penta M Di, Esposito R, Villani ML. An approach for QoS-aware service composition based on genetic algorithms. Genet Evol Comput Conf. 2005;1069.
[40] G??ne?? Kayacik H, Nur Zincir-Heywood A, Heywood M. Evolving successful stack overflow attacks for vulnerability testing. In: Proceedings - Annual Computer Security Applications Conference, ACSAC. 2005. p. 225–32.
[41] Del Grosso C, Antoniol G, Merlo E, Galinier P. Detecting buffer overflow via automatic test input data generation. Comput Oper Res. 2008;35(10):3125–43.