

A review paper: optimal test cases for regression testing using artificial intelligent techniques

Shahbaa I. Khaleel, Raghda Anan

Department of Software, College of Computer Science and Mathematics, The University of Mosul, Mosul, Iraq

Article Info

Article history:

Received Mar 26, 2022

Revised Sep 16, 2022

Accepted Oct 14, 2022

Keywords:

Fuzzy logic
Machine learning
Neural network
Regression testing
Swarm Intelligence

ABSTRACT

The goal of the testing process is to find errors and defects in the software being developed so that they can be fixed and corrected before they are delivered to the customer. Regression testing is an essential quality testing technique during the maintenance phase of the program as it is performed to ensure the integrity of the program after modifications have been made. With the development of the software, the test suite becomes too large to be fully implemented within the given test cost in terms of budget and time. Therefore, the cost of regression testing using different techniques should be reduced, here we dealt many methods such as retest all technique, regression test selection technique (RTS) and test case prioritization technique (TCP). The efficiency of these techniques is evaluated through the use of many metrics such as average percentage of fault detected (APFD), average percentage block coverage (APBC) and average percentage decision coverage (APDC). In this paper we dealt with these different techniques used in test case selection and test case prioritization and the metrics used to evaluate their efficiency by using different techniques of artificial intelligent and describe the best of all.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Shahbaa I. Khaleel

Department of Software, College of Computer Science and Mathematics, Mosul University

Mosul, Iraq

Email: shahbaaibrkh@uomosul.edu.iq

1. INTRODUCTION

Because of the rapid development in software development and the increasing need for good and efficient software systems, the importance of systems testing has increased. Through the testing process, the system is evaluated and ensured that it performs for the purpose required of it, as the testing process verifies the integrity of the program from errors and its reliability in performing its job and its correct operation [1].

There are many testing techniques and the most common are black box testing technique, white box testing technique and grey box testing technique, where black box testing focuses on the functionality of the system under development without going deep into the structural details of the system. As for the white box test, also called (structural test), it is concerned with the internal details of the system, the processes that take place within it, its internal structure and its implementation. As for grey box testing, it combines the white box testing technique and the black box testing technique. In this type of testing, the tester is familiar with the internal structure of the program and then performs functional testing of it, taking into account the internal structure of that program or application [2].

The testing process includes configuring and executing the system with test cases. Test cases are determined based on testing techniques such as black box testing, white box testing and error-based testing, through which one can know the internal structure and function of the system and some of its common errors. Testing techniques can be used in unit testing, integration testing, system testing and acceptance testing [2],

[3]. Maintenance plays an important role in the process of building and developing software. Software maintenance is the last stage of software development. The term software maintenance is used to understand the software engineering procedures that occur during the progress of the program. Software maintenance is a very complex process and usually takes more than half the time spent in the development process. The software development process usually takes from one to two years, while maintenance takes a period of up to ten years, and this depends on the reactions of users. When a certain company presents a completed project to its clients, it often starts working on maintaining these projects. It has been noted that the cost of maintenance for any project exceeds the cost of development. The maintenance phase works on adapting and updating the program with changes in the environment in which it operates, correcting errors that may appear, and improving the performance of the system after it is delivered to customers [4].

Artificial intelligence is a leading technology in the current era, among the applications of artificial intelligence that can be used to solve real-world issues are functional, analytic, interactive, visual and textual artificial intelligence to improve intelligence as well as improve application capabilities. Building and developing an effective artificial intelligence (AI) model is a challenging task due to the dynamic nature and differences in real world data and problems [5]. AI has many advantages, including [6], [7]: i) complete tasks faster than humans, ii) complete complex and stressful tasks and work easily and in a short time, iii) many tasks are accomplished simultaneously, iv) the success rate of the work is high, v) errors and defects are few and the possibility of obtaining high efficiency in a short time, vi) the size and space consumed will be small, vii) discovering undiscovered things, viii) reduce manual effort, ix) reusability of the test cases that have been created, and x) avoid redundant procedures when performing the test. This does not mean that the use of artificial intelligence is not without flaws or negatives, including: sometimes it is used wrongly, which leads to a very big mistake. Human functions are affected. Creativity depends on the programmer. His lack of a human touch. It creates a lazy generation. It requires a lot of time and budget. Increased reliance on technology. Many machine learning models and artificial intelligence techniques have been developed to build smart applications based on multimedia inputs for the purpose of achieving intelligent functional features such as recommendations, object detection, prediction, classification, and natural language processing and its translation. This results in a growing and strong demand for quality verification and assurance of AI software systems [8].

The neural networks were proposed in 1943 by McCulloch and Pitts, and in 1957 [9] the first trainable neural network called a perceptron was designed, a simple classification algorithm that has only one layer. In the 1980s in the last century, neural networks containing more than one layer were proposed to solve more complex problems such as the multi-layer perceptron (MLP) [10]. Neural networks have taken over the programming world, as they can solve tasks that are still difficult for traditional programs, as they have become an essential component of software systems that perform complex functions such as image processing, speech recognition and natural language processing, where they can reach the level of their performance to the level of human or near it [11].

There are several types of artificial neural networks, namely, feedforward neural network (FNN), recurrent neural network (RNN), and convolutional neural network (CNN). RNN is a type of short-term memory neural network that performs better than FNN in time series prediction. FNN can transmit information in only one direction. However, temporal convolutional network (TCN) which is based on CNN performs better than RNN in predicting time series [12]. The two main methods of artificial neural networks are feedforward neural networks and feedback neural networks. In feedforward neural networks, there is no feedback loops because the unit in the neural network sends information to another unit and receives nothing from it. The input and output in the feedback networks are fixed, and each unit receives the input data and information from the units located to its left, and the multiply process is performed for it with the weight of each connection, and thus the output for that connection is obtained. This method is used in network applications for which the output is known, and among these applications are classification, identification and pattern generation. In feedback neural networks, addressable contents memory is used, and the neural network is learned by comparing the network's output with the desired or expected output. The difference between the two outputs is used to change and modify the weights of connections between network units [13]. Artificial neural networks have many advantages and disadvantages [14], advantages including: adaptive learning, self-organization, real-time performance, fault tolerance. Among its disadvantages: first It can only be used when training data is available. Secondly usually the solution obtained from the learning process cannot be explained. Thirdly usually it is not possible to create a neural network using previous knowledge. Fourthly the learning process can be very long and there is no guarantee of its success.

Fuzzy logic was introduced by Zadeh in 1965 [15] and consists of a large number of probabilities or valued logic and that this logical method is very accurate and often impossible to change. Fuzzy logic variables may have one real value between 0 and 1. Fuzzy logic has been developed to reach its true value, and this value can be between completely true or completely false [14].

This research is organized as follows: The second part explains the software testing process and its importance in the software development process, including regression testing and its many techniques. The third part deals with intelligent techniques and their characteristics and examples of them. The fourth part presents a set of studies and previous works. The fifth part presents a set of comparisons between previous works included in the research. The sixth part deals with the conclusions drawn from this research.

2. SOFTWARE TESTING

The testing process plays an important role in the development of high-quality software. Testing is necessary because users and programmers all make mistakes. Some of these errors are unimportant, but others are very dangerous and expensive, so there is a need to verify all written programs [3]. The testing process means checking whether the program is free from errors and defects or not. It is a process that is useful in verifying the validity of the program and the task it performs as to whether it meets the requirements of users and what is predetermined in the required characteristics [16]. Testing means a process of comparison between the actual result and the expected result of the program. The program test is conducted to verify the correctness of the function or task performed by the system. If the test is not conducted, the system may lead to disastrous or inappropriate results. Therefore, it is better to test the system in advance excellent results can be obtained [17].

Software testing is an intrinsic and critical activity in the software development process. Its main objective is to detect the presence of errors and defects in the product under test, whether it is the program itself or its components. It is known that testing is the costliest activity in software development, which can cost greater than 50% of total cost of software development project. In addition to its main objective of detecting errors, the data collected during the testing phases is also important for debugging, maintenance, and evaluation of accuracy and reliability [18].

2.1. Regression testing

One of the basic operations in the software maintenance phase. It is the process of re-executing the test cases on the software system with its modified or improved version in order to make sure that this version is correct and not affected by the change that occurred to it, meaning that the function that it was performing in its original (previous) version has not been changed. New test cases due to the fact that the current set of test cases (belonging to the original version of the program) has become incompatible with testing the modified version and therefore new test cases are used that fit the modification or change that occurred in the program [19]. Testers may re-implement all test cases that were configured in early stages to ensure that the program works as expected, but as the size of the program increases, the test cases will grow more and it becomes difficult to retest all test cases because it will be costly in terms of time and budget. Therefore, techniques must be used to select the optimal test cases, including [3], [17]:

- a. Retest all technique: where all test cases in the test set are re-executed, but this technique is very expensive because it requires a lot of resources and time.
- b. Regression test selection technique (RTS): a specific part of the test cases in the test set is selected to be implemented, this technique has a cost less than the cost of the re-test technique.
- c. Test case prioritization technique (TCP): where the test cases are arranged appropriately to increase their effectiveness and performance and increase the rate of discovery of errors, that is, precedence is set for test cases, test cases with higher precedence are executed before test cases with lower precedence, in order to reduce time, effort and cost during the testing phase of the program. The technique of assigning precedence to test cases is classified into:
 - Customer requirement-based techniques: here, customer requirements are taken into consideration and some weights are provided, and based on these values, the weight of the test case for the requirements is evaluated. Test cases with higher value are executed first, then test cases with lower values are executed. One of the most important factors in determining the requirements of the customer is the precedence that the customer assigns to the requirements, the complexity of the requirements and the change of requirements.
 - Cost effective based technique: here, test cases that are based on cost factors such as cost of running test cases, cost of analysis, cost of setting priorities, cost of implementation and validation of test cases take precedence. The cost is divided into two types: the direct cost, which includes the selection of the test, the implementation of the test and analysis of the results, and the indirect cost, which includes the general cost and the cost of developing the tool.
 - Time based technique: In this technique, test cases that take less execution time take precedence, while test cases that take more execution time take precedence.
 - Techniques based on recording an event.

- Coverage based technique: In this technique the code coverage is analyzed and the code covered by the test case is measured, where the amount of coverage is used to assign precedence to test cases. One of the techniques that depends on the amount of coverage is the white box testing technique. Coverage is classified into several categories, including (sentence or phrase coverage, decision coverage, condition coverage and error coverage).
- Hybrid technique: In this technique, the test group reduction technique is combined with the technique of prioritizing test cases.

3. ARTIFICIAL INTELLIGENCE TECHNIQUES

In the past few years, artificial intelligence algorithms and machine learning methods have been successfully applied in many areas such as commerce, industry and digital services, and nowadays it has become widespread in software testing [20], [21]. Artificial intelligence is a branch of computer science and includes the development of computer programs to perform tasks that require human intelligence to perform. AI algorithms can process learning, cognition, problem solving, language comprehension or logical reasoning. Artificial intelligence is used in many ways in the modern world, from personal assistants to self-driving cars. Artificial intelligence is developing very quickly and science fiction portrays artificial intelligence in the form of robots that are as close as possible to humans. Artificial intelligence is also called machine intelligence, because it is intelligence demonstrated by machines in contrast to the natural intelligence shown by humans and other beings [22].

Artificial intelligence techniques have become a useful alternative to the traditional techniques used to solve complex problems in various fields and have become more widespread and popular nowadays. One of the characteristics of smart technologies is their ability to learn through examples, as well as their ability to deal with random and incomplete data, and they have the ability to deal with non-linear problems. Once the algorithm is trained, it will be able to implement prediction and generalization operations at high speed [23]. Examples of these technologies are: artificial neural networks, fuzzy logic, genetic algorithms, swarm intelligence, reinforcement learning, machine learning [24].

Soft computing is a new multidisciplinary field proposed by Zadeh [15] whose goal was to build or create a new generation of artificial intelligence called computational intelligence. It was originally created as an association or organization of computational methodologies that harness endurance energy to achieve docility, robustness, lower cost of finding solutions, and a relationship more in line with reality. Soft computing includes the following main parts: fuzzy logics, neuro computing, evolutionary computing, probabilistic computing, and swarm intelligence [25]. The reason for the success of soft computing is the collaboration and synergies that derive from its components. The main advantage of lean computing is its intrinsic ability to create hybrid systems based on the interconnected integration of these technologies. This integration provides investigative and heuristic research methods that allow the development of flexible tools for computing and solving complex problems. Soft computing applications have demonstrated two main benefits: The first is that they make it possible to solve nonlinear problems that cannot be solved by mathematical models. The second benefit is that it introduces human knowledge such as awareness, recognition, discrimination, understanding, learning and other human knowledge and integrates it with the fields and fields of computing. Figure 1 shows the components of lean computing [26].

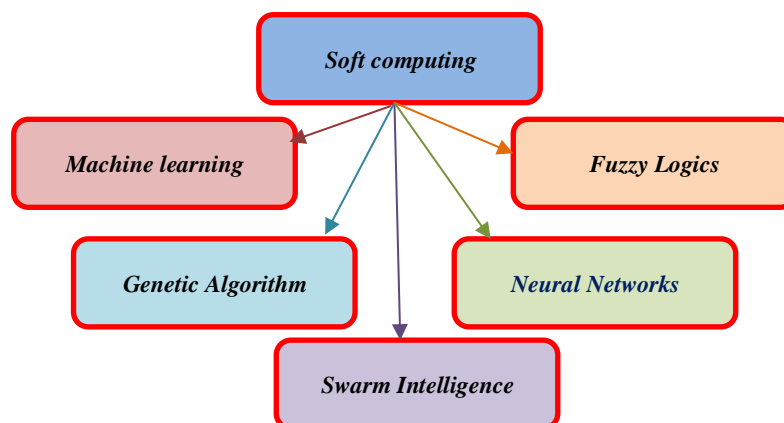


Figure 1. Contents of soft computing

3.1. Artificial neural networks

Artificial neural networks are inspired by the functions of neural networks found in the human brain. For the human brain, the signals are transmitted through a complex network of neurons, as for artificial neural networks, the inputs are converted into signals that are passed through a complex network of neurons to form outputs that can be interpreted as a response to the original input. The learning process refers to transforming the network so that these outputs are useful, intelligent, and responsive to the input. Artificial neural networks process the data that is sent to the input layer and generate a response at the output layer and in between, there is one or more hidden layers that deal with the signals as they pass through it [27].

The concept of neural networks is inspired by the biological human brain model, this concept is converted into a mathematical formulation and then turned into machine learning to solve many problems in this world. Through artificial neural networks, mathematical structures, design concepts, algorithms and computer programs can be created. Artificial neural networks have undergone many modifications in their algorithm and implementation. Otherwise, many applications contain different techniques and methods of algorithms. Various optimization techniques are used in order to get the best results depending on the problem you are solving [28]. Artificial neural networks are a type of artificial intelligence that imitates the way the human brain works and stores information. The basis of its work is to create connections between mathematical processors called neurons. The strength of network knowledge depends on the strength of connections between different cells, and these connections are called weights. A group of these neurons form a layer called layers, where the cells of this layer work in parallel and at the same time. The neural network consists of three layers: the input layer, the hidden layer, and the output layer. The system learns through the process of determining the number of neurons in each layer as well as adjust the communication weights based on the training data [29]. Artificial neural networks use many techniques in their implementation such as supervised learning, unsupervised learning and reinforcement learning [28].

3.2. Fuzzy logic

Fuzzy logic is a mathematical model that is built based on concepts taken from the theory of fuzzy set theory. It describes the system by establishing the existing relationships between the inputs and the outputs in the form of certain rules [30]. The idea of fuzzy groups and fuzzy logic was put forward by the Azerbaijani scientist Zadeh [15], who developed a linguistic method for dealing with ambiguous linguistic information based on fuzzy groups and fuzzy logic. Since then, this method has been used in practical applications in various fields, including medical, industrial, engineering, meteorological sciences, digital image processing, business administration, computer science, expert systems and others [25].

Fuzzy groups differ from traditional groups in that their boundaries are imprecise, as an element can be partial to the fuzzy group. The concept (warmth), for example, includes a wide range of measured temperatures with a specific scale, this concept can be considered as a name for the foggy group, and then any measured temperature can belong to this group partially, that is, it is confined within the period [0,1] where one means full belonging to the fuzzy group and zero means not belonging to the group. The fuzzy model is an expert system that clarifies the relationship between inputs and outputs through a set of rules. Building the fuzzy pattern usually goes through three basic steps, which can be illustrated by Figure 2 [30].

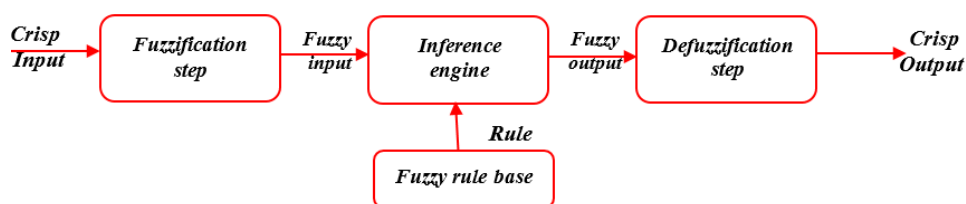


Figure 2. Stages of a fuzzy logic system

3.3. Genetic algorithm

The genetic algorithm was discovered by the scientist Holland [31] at the University of Michigan. The term genetic algorithm is taken from embryology, because the genetic algorithm is based on an idea taken from Darwin's theory of evolution, new society [32]. The genetic algorithm is a search algorithm based on the principle of natural selection, where the solution starts with a population with random values representing a set of solutions, and each solution has a specific fitness function that is directly related to the goal function of the particular issue and then this community is modified and another new generation is generated through the application of a set of genetic processes, including selection, crossover, and mutation, repeatedly and sequentially on the individuals of the generation until the stopping condition is fulfilled [33].

Genetic algorithms are nowadays cleverly used in engineering as an adaptive technique for solving complex problems and issues. The genetic algorithm uses selection, crossover and mutation operations to effectively manage the search system's strategy. This algorithm is derived from the concepts of natural and genetic selection, as well as uses random search supported by historical data (heuristic data) to contribute to the search within a set of improved results within a certain framework. Some algorithms are widely used to maintain the best feedback in order to improve some tasks and investigate problem solving [34].

The genetic algorithm is good for some works that require optimization. It is applied to problems that have a large area and large variables and can be solved easily and quickly. It also gives a solution that is very close to the ideal solution, and most likely there are a number of possible solutions and one of these solutions is the best or optimal [35].

3.4. Machine learning

It is one of the most useful research areas at the moment, whether in proposing new techniques or theoretical algorithms and applying them to real-life problems. The world has changed at an unexpected pace and as a result it has become possible to use high-quality and fast devices at a relatively cheap price. The development of systems that can adapt to the environment in a smart way has a very large field of work and the most important characteristic in the field of machine learning is that it collects knowledge from different fields, for example in the field of pattern recognition can combine neural networks, support machine, decision tree, and other fields, so it benefits from the synergy between all of these fields, and then provides effective solutions that are used in various fields of knowledge [36]. Machine learning has become an important area nowadays because of its importance to development organizations that are looking for innovative ways to take advantage of original data and make business more understandable.

Machine learning is computational methods that use experience in order to improve performance or to make accurate predictions, and experience here means the previous information available to the learner, which usually takes the form of electronic data collected and made available for the purpose of analysis, and these data can be in the form of sets of digital data which are related to human signs or may be information obtained through interaction with the environment. In all cases, quality and size are necessary and fundamental to the success of the predictions made by the learner [37]. In recent years, researchers have relied on machine learning techniques to achieve effective reduction and selection of test cases and the processes of assigning precedence to test cases. These techniques help in collecting information about test cases from partial and incomplete sources to obtain accurate and effective prediction models [38].

3.5. Swarm intelligence

It is one of the techniques of artificial intelligence and inspired by the collective behavior of societies such as animal societies. Each individual in the group is independent. It is a sub-system that interacts with its environment, which consists of other individuals to form an integrated system. This individual does not comply with certain orders from a leader or a general plan followed by all individuals, for example, a bird in a flock of birds is independent of its decisions, but it adjusts its movement in a manner consistent with the movement of other birds in the same flock and tries to stay next to the birds that walk near it and avoid collision with them. This process has been proven to increase the collective intelligence of these societies, such as flocks of fish, flocks of birds and flocks of bees [39].

In general, all methods of swarm intelligence depend on the community, where in this community there are a set of potential solutions, the ideal solution is searched through iterative steps, and the members of the community change their positions within the search space through vector and non-vectorized connections [40]. Swarm intelligence describes the ability of groups of animals and insects to exhibit highly organized behaviors to solve complex problems that allow the group as a whole to accomplish tasks that are beyond the ability of the single individual in the group. This natural phenomenon is the inspiring factor for swarm intelligence systems [41]. In computational terms, they match the naturally occurring behavior of an insect community or swarm in order to simplify the design of distributed solutions to complex problems [42].

4. PREVIOUS WORKS

Khanna [43] proposed a technique used to set precedence [43] for test cases that uses neural networks neural networks (NN) to rank test cases where the NN is trained to assign the precedence of test cases to a regression test suite. Initially, the NN is provided with test cases and rules to assign precedence to them, and then the NN is allowed to learn the techniques for assigning precedence and the rules used to set this precedence. This learned NN is verified through a set of test cases. During verification and testing phase, the input to the NN that it is trained is a set of test cases and the output to the NN is the test cases that have been ordered and given precedence.

Hooda and Chhillar [44] presented a new concept using UML state diagrams and tables to create test cases and used artificial neural networks as an improvement tool by reducing duplicate test cases generated using the genetic algorithm. The neural network was trained using a feedback algorithm for a set of test cases that were executed on the original version of the program. From the results, it became clear that the algorithm provides maximum efficiency and coverage for the code, as well as good handling of iterations in the generated test cases.

Wang *et al.* [45] a test case selection algorithm that can automatically select a high-coverage but small-sized test set when the budget is low or limited. They also suggested an algorithm to assign priority to test cases. Extensive experiments were carried out on two datasets and four common models of deep neural network (DNN). The results showed that the algorithm can achieve higher coverage with small test sets than DeepXplore (used as the basis and is a method for generating a mutation test set), and the experiments also showed that the prioritization algorithm is able to find more errors. It achieved a recall rate of 94.1%.

Lousada and Ribeiro [46] used neural networks to assign test cases priority to reduce the cost of regression testing, depending on the complexity of the modules that were modified. The parts of the program interact with each other and that any change in one part will affect other parts, so a matrix was created for the interaction of the parts of the program and a map was created for the amount of coverage of the test in order to test the interaction between these parts. Each intersection in the matrix between the interactions of the part of the program represents the degree of complexity for those interacting units, this complexity has five different dimensions: data, logic, environment, structure, and use case. A feedforward neural network has been built and its input is the five complexity dimensions and its output is the test case number with a value of either 0 or 1 and the dimensions are arranged from 1 to 5 where the number 1 is given to the least affected test cases and the number 5 is given to the test cases that have great reliability (affected more) and depending on this arrangement, a data set is formed for different groups of possible inputs and outputs for each interaction belonging to a part of the program.

Kaur and Goyal [47] adopted code coverage in assigning precedence to test cases and used the genetic algorithm. The results showed the efficiency of the genetic algorithm in assigning precedence to test cases based on the average percentage of code coverage (APCC) metric. Suman and Seema [48] used a new genetic algorithm to assign precedence to a set of test cases, which dynamically arranges test cases based on the amount of full code coverage, and at the same time introduces a method for generating new test cases using partially mapped crossover (PMX) and cyclic crossover. The analysis is done based on the cost of this process and the cost of testing. The overall goal of their research is to reduce the number of test cases that must be run after maintenance and program changes. They concluded that this algorithm can be used effectively if there are a large number of test cases and a large number of possible test sequences, and this work resulted in an ideal test sequence.

Mishra and Pattaniak [49] introduced a new technique for assigning precedence to test cases using a genetic algorithm that isolates test cases that are detected as severe by the customer and from among the remaining test cases gives precedence to the original test set so that the new set is executed in an environment execution specific time. This algorithm has a high error detection rate when compared with test groups with random precedence. In their research, they analyzed the genetic algorithm in terms of efficiency and time by using a structure-based criterion to assign precedence to test cases. The test cases generated by assigning precedence to goal oriented, random and path-oriented test cases were compared and found some better results in terms of time and code coverage.

Singh and Kau [50] contained their experience using a genetic algorithm to assign precedence to test cases based on their rate of detected errors and execution time. The results showed the efficiency of this algorithm, as the efficiency and performance of the work were evaluated using the APCC metric. Khanna [51] in which a genetic algorithm was used to assign precedence to test cases, based on the rate of detected errors, as well as relied on several methods to set priorities, which are the amount of coverage, severity of risk, cost and customer requirements. Khanna analyzed it based on fault coverage and applied this to multiple programs.

Bajaj and Sangwan [52] applied the genetic algorithm to a reality program and used the average percentage of fault detected (APFD) metric to assess the efficiency of assigning precedence to test cases. And it was proved that the cycle selection system has better performance compared to other test systems such as the roulette wheel or the random system. With fitness function evaluations, and thus their values affect the final result and the time taken, that is, they should not be too little or too high.

Bhasin *et al.* [53] relied on the ability of the expert fuzzy system to make decisions as they based their technology on the coverage and effect of the error or defect in selecting the test case. This work improved and enhanced the technology based on existing fuzzy logic and eliminated the defects in it. The proposed architecture of the fuzzy regression expert system (FRES) consists of three components which are the knowledge base, the inference engine and the user interface. The knowledge base contains all the rules, and the inference engine makes the decision by checking the rules that check the facts, and the inference engine works on arranging and giving priority to the rules that fulfill the highest priority rules. As for the user

interface, it presents the available facts related to the user as well as other information such as input. The proposed work has been tested using an economical program consisting of 8,000 lines of code as well as about 200 modules and 500 test cases. In their research, they found that the use of expert fuzzy logic gives better results than other decision-making systems, as the proposed work was analyzed and implemented, and the results obtained were very encouraging.

Ghai and Kaur [54] proposed a technique that traverses the data flow diagram of the project or system and arranges the precedence of test cases according to their functional importance, which is calculated using automated slicing, where the functional importance values are given as inputs to the hill-climbing algorithm, which in turn gives Precedence for test cases in ascending or descending order according to functional importance. They implemented the algorithm using MATLAB language and analyzed the increase in the value of error detection by up to 20%, and concluded that this technique performed well in detecting the error.

Mece *et al.* [55] that included studies on the use of machine learning applications to assign precedence to test cases, metrics used to measure the efficiency of these machine learning methods, and data used to assign precedence to test cases. In their research, they explained the pros and cons of machine learning applications in the field of assigning precedence to test cases TCP, as there are several machine learning techniques used for this purpose, such as NN, Bayesian networks and genetic algorithm (GA). Through studies and comparisons, they concluded that NN is good in terms of classification (good classifier), and Bayesian networks and GA are good in terms of ordering test cases, and that NN and Bayesian networks are good in terms of adaptation as they can they include many different features, and that GA and Bayesian networks are very easy to implement, they also concluded that machine learning techniques are very effective in the matter of assigning precedence to test cases because of their ease of adaptation to changes and because they give good results in assigning precedence and classification despite its disadvantages as it requires many data sets for the training process.

Kaur and Bhatt [56] aimed at detecting errors in the shortest possible time and used the hybrid swarm optimization (HPSO) algorithm to make the regression test more efficient. This algorithm is a combination of the improved swarm algorithm and the genetic algorithm in order to expand the search space for the solution. This combination of the two algorithms provides a quick solution and makes it more efficient. The mutation operator was used, which in turn allows the search engine to evaluate all aspects of the search space. The metric APFD was used to represent the solution derived from the algorithm HPSO in order to obtain better transparency for this algorithm.

Suri and Mangal [57] introduced a tool called HBG_TCS, which they developed using C++. This tool implements the proposed hybrid algorithm, which depends on the improved bee colony algorithm and the genetic algorithm in reducing the number of test cases. This algorithm takes test cases, errors and execution time as input. This results in a minimal set of test cases. In their research, they evaluated the validity and efficiency of this tool and compared it with another tool based on the improved ant colony algorithm. They concluded that the tool HBG_TCS, which depends on the hybrid technology (GA + BCO) in selecting test cases, was faster than the tool ACO_TCSP, which depends on the improved ant colony algorithm. That is, the implementation time has been greatly reduced, and thus the cost of the regression testing has been reduced.

Bajwa and Kaur [58] proposed the use of a hybrid technology, which is a combination of an adaptive method and a genetic algorithm. A good rate of speed is obtained by using the adaptive method, which schedules the test cases simultaneously during the execution of these cases. At first, the adaptive method is used to determine the precedence of the test cases by choosing the test case with the highest precedence, then the test cases are executed and their output is recorded. Relying on this output and on the execution log of the next unselected test case, the next test case takes precedence. This process ends when all test cases covering the entire code statements have been arranged and executed. The genetic algorithm arranges the test cases using four operations: parents testing, crossover, mutation, and duplication elimination.

Sachdeva [59] made a comparison between the performance of the swarm algorithm with the genetic algorithm in the year 2020 in order to assign precedence to test cases by comparing between the ant colony optimization (ACO) algorithm and the GA in order to reduce the set of test cases and the comparison was based on execution time by comparing the results with the time taken for both of them randomly. The results showed that the GA and the ACO algorithm are much better than other traditional techniques in determining precedence. In addition, the execution time of the genetic algorithm is very close to the execution time of the ACO algorithm, while other testing methods take five times more time or more. The ACO algorithm and the GA have the same drawbacks, so it is preferable to use the GA with the bee colony optimization algorithm (BCO) or another algorithm in addition to the ACO algorithm in order to complement one another and improve the ability of the test group because using a hybrid method leads to better performance than using swarm algorithms alone.

Sethi *et al.* [60] used an improved ant colony algorithm to reduce the number of test cases and assign them precedence. Their main goal was to get the shortest path and solve the time problem, i.e., reduce the time to find the shortest path. They used their algorithm to select the test case and set the precedences based on the amount of error coverage and execution time, allowing the tester to set the test case precedence without taking into account the number of lines covered by the test case, which might contain an extra comment line. They improved the ant colony algorithm and used the APFD metric to compare the efficiency of the improved algorithm with the original algorithm, and they proved that the modified technique gives better and more effective results than the original technique.

Ansari *et al.* [61] proposed a technique for prioritizing regression test cases that automatically optimizes the test set to be more efficient. They used an ant colony optimization algorithm, where a set of test cases are taken as input and arranged based on a specific criterion, thus improving the efficiency of the test process. In the beginning, a test case is chosen that covers the largest possible number of errors, because the goal is to detect the error as much as possible, and it is examined whether it covers all errors or not. If not, the next test case is selected, which covers the remaining errors, and it is repeated until all errors are detected, then the total number of errors detected for each test case is calculated, which is stored in an array of all errors for the test cases. This leads to the so-called paths that cover all errors and are several of these paths are explored during the iteration and the best path is determined and the one that takes the least execution time. They concluded that the implementation of this technique will reduce the time, effort and cost in the testing process and detect errors as much as possible. They used the metric APFD to evaluate the efficiency of the algorithm.

Ashraf *et al.* [62] proposed using a swarm algorithm to assign precedence to test cases. They introduced six previously unused factors (client precedence, changes in requirements, implementation complexity, requirements tracking, execution time, and impact of requirements error). In their research, they applied the algorithm to three medium-sized programs, and compared the results with a random technique to assign precedence, and it turned out that the used algorithm was the most efficient and powerful in detecting errors in the advanced stages of the system building cycle. They used the APFD metric in efficiency evaluation.

Kaur and Agrawal [63] presented a paper in which they evaluated the performance of two algorithms for selecting test cases, the Bat algorithm and the Cuckoo search algorithm, and their evaluation was based on two factors: execution time and the number of errors detected. The results showed that the Cuckoo search algorithm was better than the bat algorithm in detecting errors. In terms of execution time, the bat algorithm was better than the cuckoo search algorithm.

Ahmad *et al.* [64] presented a hybrid method of intelligent techniques, which is a combination of the ant colony algorithm and the genetic algorithm, where the test cases are initially created based on their precedence, where their precedence is determined based on specific test factors such as importance, variability, complexity, percentage error, time, and amount of coverage. Then the sequence with the lowest execution time and highest error rate is calculated by the ant colony optimization algorithm.

Dhareula and Ganpati [65] used the cuckoo search algorithm (CSA) to assign precedence to test cases. Where CSA divides test cases into groups depending on the amount of code coverage for each test case, three groups were used to implement the CSA, these groups contain a different set of test cases in different arrangements and the test cases in each group are determined based on their code coverage, they adopted on the APFD metric in verifying the validity of the results by comparing the performance of the flower pollination algorithm (FPA) and the traditional methods of assigning precedence to test cases, and they used the Java language (JAVA) with the Eclipse platform in implementing the CSA on two different programs, and the results showed that the CSA was superior to the FPA when both programs were implemented.

Manaswini and Reddy [66] applied cat swarm optimization algorithm for ordering test cases to assign precedence to these cases based on the fitness function value, which is calculated using some metrics such as coverage amount, error detection rate and execution time for many test cases, they implemented the algorithm on open source software such as jtopans and jmeter. The algorithm included two steps, the first is seeking and the second is tracking, which helps in arranging the test cases in a specific order depending on the fitness function. The results showed the efficiency of the algorithm in terms of execution time as well as in terms of error detection rate.

Khatibsyarbini *at al.* [67] proposed an approach to optimally assigning precedence to test cases using the Firefly algorithm to improve the order of test cases, where they applied this algorithm with a fitness function defined using similarity distance model. Experiments were conducted on three programs and the efficiency and effectiveness of this algorithm was verified in determining the precedence of test cases. The results showed that this algorithm achieved good results when evaluating its efficiency using the metric average percentage of fault detected (APFD), as well as outperforming this algorithm over local beam search (LBS) in terms of execution time.

Vats and Kumar [68] in which they applied a cat swarm optimization (CSO) algorithm to assign precedence to test cases. The error detection rate is one of the most important criteria for the efficiency of the cat swarm algorithm. The algorithm work included two steps, the first step is to identify different groups of test cases based on a test suite and the second step is to evaluate the performance of the test cases. The results showed that the CSO algorithm achieves good results compared to other algorithms, it is also considered an APFD error detection measure.

Reda *et al.* [69] who used the multi-objective adapted binary bat algorithm (MO-ABBA) to solve the test suite reduction (TSR) problem. This improved algorithm was evaluated based on eight programs with different sizes of test groups and twelve benchmark functions and based on five measures, namely Mean, standard deviation (SD), test suite size reduction rate (TSRR), execution time reduction rate (ETR), fault detection capability rate (FDR), it was found that MO-ABBA significantly reduces the number of test cases compared to the original algorithm before optimization multi-objective binary bat algorithm (MO-BBA) as well as more than the MO-BPSO.

Sharma and Rastogi [70] suggested in her research a technique to reduce test cases and assign precedence to them. This technique or algorithm works in two stages. In the first stage, the selection process for test cases is performed, and in the second stage, priority is set for the tested test cases depending on the severity of error detection. When comparing the efficiency of this technique with the efficiency of other techniques used for the same purpose, the results showed that the value of the APFD metric is greater than its value in the absence of arranging the test cases or in the case of random arrangement, but its value is less than its value if the system is arranged using the error rate algorithm per minute.

Pravin and Srinivasan [71] used a technique to reduce the set of test cases and test a subset of them that will be executed in the shortest time and have the ability to cover all errors and then rearrange the lower case on the basis of a time fault. The precedence assigning technique arranges the test cases using an error detection algorithm. After errors are found in the code, the source code will be processed in an open-source system such as WebKit. The main objective of the error detection algorithm is to reduce the random selection of the test case by assigning the same values to two errors. The efficiency of this technique has been evaluated using the APFD metric using different values of time, and the results showed when comparing the number of test cases and efficiency that the greater the number of test cases, the efficiency and the ability to detect error also increases, and when comparing the existing techniques with the technology assumed in this research, the value of fault detected (FD) (which is a measure of error detection of the technique used) was greater than the value of APFD, meaning that FD has the ability to detect error more than APFD.

Beena and Sarala [72] suggested using a method for selecting and assigning precedence to test cases to obtain the largest amount of code coverage. Where, in the first stage, an algorithm is used to select test cases from the test set, and in the second stage, an algorithm is used to assign precedence to the test cases that were selected in the previous stage. When comparing the number of test cases before using the selection algorithm and their number after using the algorithm, the results showed that their number had decreased significantly, and the results also showed that after using the priority assigning algorithm, the number of test cases had decreased very significantly, thus reducing the cost of regression testing and the time of executing test cases to a large extent.

Dahiya *et al.* [73] compare between the techniques for assigning precedence to test cases. Possible sentence, phrase or function, the ability of the technology to take into account the risks related to the test cases) and they compared the techniques on the basis of the extent to which they achieve these objectives. They concluded that each method or technology has benefits and limitations according to the requirements of the testing process, the size of the program, as well as the requirements and test environment. The goal is always to choose a technology that reduces cost and increases the rate of error detection as quickly as possible. Based on the previous works above, Table 1 has been organized to make a comparison between them to clarify what are the smart techniques used to conduct the testing process on the programs, with a mention of the metrics used in these works to measure the quality and efficiency of the method.

Through the Table 1, it became clear that when swarm intelligence techniques are used, among the smart techniques, they give the best results, whether in terms of obtaining the best test cases and the best test data, or in terms of the time taken to examine the program, no matter how high or simple its complexity is, so we recommend adopting swarm techniques In solving the problem of software testing, finding the best paths and best test cases, as well as cross-breeding swarm intelligence algorithms with other algorithms, or using coverage criteria for regressive testing as a function of fitness within swarm algorithms. In addition to the foregoing of using intelligent algorithms to conduct the testing process for programs also within the regressive test to discover errors within the programs, the algorithm for selecting test cases and the method of reducing test cases was used, as well as the precedence of test cases, as it became clear from previous work that when using these methods together leads to the best results.

Table 1. The intelligent methods used in the testing process

Authors	Year	Technique used	Regression testing technique	Efficiency metric used
Khanna [43]	2015	NN	Test case prioritization	-----
Hooda and Chhillar [44]	2018	NN	Reducing duplicate test cases	-----
Wang <i>et al.</i> [45]	2020	DNN	Test case selection	recall rate
Lousada and Ribeiro [46]	2020	NN	Test case prioritization	average percentage of transition detection (APTD)
Kaur and Goyal [47]	2011	GA	Test case prioritization based on code coverage	APCC
Suman and Seema [48]	2012	GA	Test case prioritization based on code coverage	Cost
Mishra and Pattaniak [49]	2013	GA	Test case prioritization based on error severity, code coverage and time	Execution time (0.0312 to 0.9958 s), Code coverage
Singh and Kau [50]	2014	GA	Test case prioritization based on error detection rate and time	APCC
Khanna [51]	2016	GA	Test case prioritization based on error detection rate	Execution time
Bajaj and Sangwan [52]	2019	GA	Test case prioritization based on error detection rate	APFD
Bhasin <i>et al.</i> [53]	2013	Fuzzy logic	Test case selection	Defect, impact and coverage
Ghai and Kaur [54]	2017	machine learning (ML)	Test case Prioritization based on functional importance	Error detection rate
Mece <i>et al.</i> [55]	2020	ML	Test case prioritization based on model, coverage rate, fault rate	APDF
Kaur and Bhatt [56]	2011	Swarm & GA	Test case prioritization based on time	APFD
Suri and Mangal [57]	2012	Hybrid (swarm/BCO & GA)	Test case selection	Execution time (2-7 s)
Bajwa and Kaur [58]	2017	Hybrid (Adaptive & GA)	Test case prioritization based on code coverage and time	Execution time
Sachdeva [59]	2020	Swarm & GA	Test case prioritization based on time	Execution time GA (1.06 to 3.4 ms) ACO (1 to 2.98 ms) Random testing (8 to 19.66 ms)
Sethi <i>et al.</i> [60]	2014	Swarm/ACO	Test case prioritization based on error detection rate and execution time	APFD
Ansari <i>et al.</i> [61]	2016	Swarm/ACO	Test case prioritization based on time	APFD
Ashraf <i>et al.</i> [62]	2017	Particle Swarm	Test case prioritization based on error detection rate	APFD
Kaur and Agrawal [63]	2017	Swarm/Bat & Cuckoo	Test case selection	Error detection rate, Execution time Bat (0.19 to 1.81 s) Cuckoo (0.15 to 1.56 s)
Ahmad <i>et al.</i> [64]	2018	Swarm/ACO	Test case prioritization based on importance, volatility, complexity, fault rate, time and coverage	Execution time, Error detection rate
Dhareula and Ganpati [65]	2019	Swarm/Cuckoo	Test case prioritization based on code coverage	APFD, Execution time Prog1 (0.255 to 0.94 s) Prog2 (0.078 to 0.792 s)
Manaswini and Reddy [66]	2019	Swarm/Cat	Test case prioritization based on error detection rate and time	Error detection rate, Execution time: (2 to 9 s)
Khatibsyarhini <i>at al.</i> [67]	2019	Swarm/Firefly	Test case prioritization based on time	APFD
Vats and Kumar [68]	2020	Swarm/Cat	Test case prioritization based on error detection rate and time	APFD
Reda <i>et al.</i> [69]	2021	Swarm/Bat	Test case minimization	Mean, SD, TSRR, ETR, FDR
Sharma and Rastogi [70]	2013	Fault Severity	Test case reduction and prioritization based on error severity	APFD
Pravin and Srinivasan [71]	2013	Fault detection algorithm	Test case selection	APFD
Beena and Sarala [72]	2013	Selection and prioritization technique	Test case reduction and prioritization based on code coverage	Test suite size

5. CONCLUSION

Since developments oriented towards testing are very profitable, testing is no longer an additional activity. Techniques for assigning precedence to test cases actually help in the orderly execution of test cases based on the performance or function of the target (amount of coverage, execution time, and cost). In this study, a survey was made about regression testing and the techniques used in selecting the optimal test cases, and a group of previous works was addressed and compared among them. After reading all the selected research in full, the following was concluded: First, from the review of the works, it was found that the APFD metric is the most widely used in determining the precedence of test cases, as well as the execution time and the amount of error coverage are largely used as a measure for evaluating test cases. Second, the technique of assigning precedence based on the amount of error coverage and time is the dominant technique in assigning precedence to test cases, followed by dependence on the amount of code coverage. Also, it became clear from the previous works that the methods and techniques of artificial intelligence used to conduct the testing process on the programs had varying performance among them and that the methods of swarm intelligence were the best and the best results were obtained by adopting the hybrid methods of swarm intelligence.

ACKNOWLEDGEMENTS

Authors would like to thank the University of Mosul in Iraq for providing Moral support.

REFERENCES




- [1] M. A. Umar, "Comprehensive study of software testing: categories, levels, techniques, and types," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 5, no. 6, pp. 32–40, 2019, doi: 10.36227/techrxiv.12578714.
- [2] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," in *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, Nov. 2016, pp. 177–182, doi: 10.1109/ICT4M.2016.045.
- [3] S. Khaleel and R. Khaled, "Selection and prioritization of test cases by using bee's colony," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 11, no. 1, pp. 179–201, Jul. 2014, doi: 10.33899/csmj.2014.163746.
- [4] A. Gupta and S. Sharma, "Software maintenance: challenges and issues," *International Journal of Computer Science Engineering (IJCSE)*, vol. 4, no. 1, pp. 23–25, 2015.
- [5] I. H. Sarker, "AI-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems," *SN Computer Science*, vol. 3, no. 2, Mar. 2022, doi: 10.1007/s42979-022-01043-x.
- [6] A. Trudova, M. Dolezel, and A. Buchalceva, "Artificial intelligence in software test automation: A systematic literature review," in *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering*, 2020, pp. 181–192, doi: 10.5220/0009417801810192.
- [7] K. C. A. Khanzode and R. D. Sarode, "Advantages and disadvantages of artificial intelligence and machine learning: A literature review," *International Journal of Library & Information Science (IJLIS)*, vol. 9, no. 1, pp. 30–36, 2020.
- [8] J. Gao, C. Tao, D. Jie, and S. Lu, "Invited paper: What is AI software testing? and why," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Apr. 2019, pp. 27–2709, doi: 10.1109/SOSE.2019.00015.
- [9] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943, doi: 10.1007/BF02478259.
- [10] J. Zhang and J. Li, "Testing and verification of neural-network-based safety-critical control software: A systematic literature review," *Information and Software Technology*, vol. 123, Jul. 2020, doi: 10.1016/j.infsof.2020.106296.
- [11] M. Rezaalipour and C. A. Furia, "Test-case generation for finding neural network bugs," *arXiv:2112.05567*, Dec. 2021. [Online]. Available: <http://arxiv.org/abs/2112.05567>.
- [12] H. Xiao, M. Cao, and R. Peng, "Artificial neural network based software fault detection and correction prediction models considering testing effort," *Applied Soft Computing*, vol. 94, p. 106491, Sep. 2020, doi: 10.1016/j.asoc.2020.106491.
- [13] R. Dastres and M. Soori, "Artificial neural network systems," *International Journal of Imaging and Robotics*, vol. 21, no. 2, pp. 13–25, 2021.
- [14] Z. Pezeshki and S. M. Mazinani, "Comparison of artificial neural networks, fuzzy logic and neuro fuzzy for predicting optimization of building thermal consumption: a survey," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 495–525, Jun. 2019, doi: 10.1007/s10462-018-9630-6.
- [15] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965, doi: 10.1016/S0019-9958(65)90241-X.
- [16] A. Uddin and A. Anand, "Importance of software testing in the process of software development," *IJSRD - International Journal for Scientific Research & Development*, vol. 6, no. 12, pp. 141–145, 2019.
- [17] M. Qasim, A. Bibi, S. J. Hussain, N. Z. Jhanjhi, M. Humayun, and N. U. Sama, "Test case prioritization techniques in software regression testing: An overview," *International Journal of Advanced and Applied Sciences*, vol. 8, no. 5, pp. 107–121, May 2021, doi: 10.21833/ijaas.2021.05.012.
- [18] S. Khaleel and A. Al Thanoon, "Design a tool for generating test cases using swarm intelligence," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 10, no. 1, pp. 421–444, Mar. 2013, doi: 10.33899/csmj.2013.163468.
- [19] M. Al-Refai, W. Cazzola, and S. Ghosh, "A fuzzy logic based approach for model-based regression test selection," in *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Sep. 2017, pp. 55–62, doi: 10.1109/MODELS.2017.17.
- [20] R. Lima, A. M. R. da Cruz, and J. Ribeiro, "Artificial intelligence applied to software testing: A literature review," in *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, Jun. 2020, pp. 1–6, doi: 10.23919/CISTI49556.2020.9141124.
- [21] B. Khaleel, "Image clustering based on artificial intelligence techniques," *AL-Rafidain Journal of Computer Sciences and*

- Mathematics*, vol. 11, no. 1, pp. 99–112, Jul. 2014, doi: 10.33899/csmj.2014.163735.
- [22] Z. Mohammed, *Artificial intelligence definition, ethics and standards*. StuDocu, 2019.
- [23] A. Mellit, S. A. Kalogirou, L. Hontoria, and S. Shaari, “Artificial intelligence techniques for sizing photovoltaic systems: A review,” *Renewable and Sustainable Energy Reviews*, vol. 13, no. 2, pp. 406–419, Feb. 2009, doi: 10.1016/j.rser.2008.01.006.
- [24] L. F. de Lima, L. M. Peres, A. R. A. Gregio, and F. Silva, “A systematic literature mapping of artificial intelligence planning in software testing,” in *15th International Conference on Software Technologies (ICSOFT 2020)*, 2020, pp. 152–159.
- [25] B. Khaleel, “Color image retrieval based on fuzzy neural networks and swarm intelligence,” *IJUM Engineering Journal*, vol. 23, no. 1, pp. 116–128, Jan. 2022, doi: 10.31436/iujmej.v23i1.1802.
- [26] S. Khaleel, “Image compression using swarm intelligence,” *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 1, pp. 257–269, Feb. 2021, doi: 10.22266/ijies2021.0228.25.
- [27] P. Boucher, “Artificial intelligence: How does it work, why does it matter, and what can we do about it?,” *European Parliamentary Research Service, Scientific Foresight Unit (Stoa)*, pp. 1–76, 2020.
- [28] A. G. Farizawani, M. Puteh, Y. Marina, and A. Rivaia, “A review of artificial neural network learning rule based on multiple variant of conjugate gradient approaches,” *Journal of Physics: Conference Series*, vol. 1529, no. 2, Apr. 2020, doi: 10.1088/1742-6596/1529/2/022040.
- [29] S. Khaleel and K. Mahdi saleh, “Detection of network anomaly based on hybrid intelligence techniques,” *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 9, no. 2, pp. 81–98, Dec. 2012, doi: 10.33899/csmj.2012.163720.
- [30] D. Ibrahim, “An overview of soft computing,” *Procedia Computer Science*, vol. 102, pp. 34–38, 2016, doi: 10.1016/j.procs.2016.09.366.
- [31] J. H. Holland, “Genetic algorithms and adaptation,” in *Adaptive Control of Ill-Defined Systems*, Boston, MA: Springer US, 1984, pp. 317–333.
- [32] B. I. Khaleel, “Butterflies image recognition and classification based on meta-heuristic algorithms,” *Journal of Engineering Science and Technology*, vol. 17, no. 3, pp. 1985–1999, 2020.
- [33] B. Khaleel, “Using artificial intelligence techniques for image compression,” *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 11, no. 2, pp. 65–81, Dec. 2014, doi: 10.33899/csmj.2014.163750.
- [34] T. Alam, S. Qamar, A. Dixit, and M. Benaida, “Genetic algorithm: Reviews, implementations, and applications,” *International Journal of Engineering Pedagogy (IJEP)*, vol. 10, no. 6, 2020, doi: 10.36227/techrxiv.12657173.
- [35] S. I. Khaleel, “Image edge detection based on swarm intelligence,” *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 6, pp. 321–332, Dec. 2021, doi: 10.22266/ijies2021.1231.29.
- [36] B. I. Khaleel and M. Y. Ahmed, “Pneumonia detection using butterfly optimization and hybrid butterfly optimization algorithm,” *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2037–2045, Aug. 2021, doi: 10.11591/eei.v10i4.2872.
- [37] M. Mohr, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. The MIT Press, 2012.
- [38] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, “Test case selection and prioritization using machine learning: a systematic literature review,” *Empirical Software Engineering*, vol. 27, no. 2, Mar. 2022, doi: 10.1007/s10664-021-10066-6.
- [39] I. Obagbuwa, “Swarm intelligence algorithms and applications to real-world optimization problems: A survey,” *International journal of simulation: systems, science & technology*, vol. 19, no. 2, pp. 51–58, May 2018, doi: 10.5013/IJSSST.a.19.02.05.
- [40] S. I. Khaleel and R. W. Khaled, “Image retrieval based on swarm intelligence,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 6, pp. 5390–5401, Dec. 2021, doi: 10.11591/ijece.v11i6.pp5390-5401.
- [41] S. Khaleel, “Designing a tool to estimate software projects based on the swarm intelligence,” *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 524–538, Aug. 2021, doi: 10.22266/ijies2021.0831.46.
- [42] B. I. Khaleel and M. S. Aziz, “Using artificial intelligence methods for diagnosis of gingivitis diseases,” *Sixth International Scientific Conference for Iraqi Al Khwarizmi Society (FISCAS)*, vol. 1897, 2021.
- [43] E. Khanna, “Neural network based regression testing,” *International Journal of Advanced Technology in Engineering and Science*, vol. 3, no. 1, pp. 133–138, 2015.
- [44] I. Hooda and R. S. Chhillar, “Test case optimization and redundancy reduction using GA and neural networks,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 5449–5456, Dec. 2018, doi: 10.11591/ijece.v8i6.pp5449-5456.
- [45] Z. Wang, S. Xu, X. Cai, and H. Ji, “Test input selection for deep neural networks,” *Journal of Physics: Conference Series*, vol. 1693, no. 1, Dec. 2020, doi: 10.1088/1742-6596/1693/1/012017.
- [46] J. Lousada and M. Ribeiro, “Neural network embeddings for test case prioritization,” *arXiv:2012.10154*, Dec. 2020, [Online]. Available: <http://arxiv.org/abs/2012.10154>.
- [47] A. Kaur and S. Goyal, “A genetic algorithm for regression test case prioritization using code coverage,” *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 3, no. 5, pp. 1839–1847, 2011.
- [48] Suman and Seema, “A genetic algorithm for regression test sequence optimization,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 1, no. 7, pp. 478–481, 2012.
- [49] P. Kumar Mishra and B. K. S. S. Pattanaik, “Analysis of test case prioritization in regression testing using genetic algorithm,” *International Journal of Computer Applications*, vol. 75, no. 8, pp. 1–10, Aug. 2013, doi: 10.5120/13128-0484.
- [50] K. Singh and P. Kau, “Efficient test cases of regression testing using genetic algorithm,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 7, pp. 7504–7506, 2014.
- [51] E. Khanna, “Regression testing based on genetic algo,” *International Journal of Computer Applications*, vol. 154, no. 8, pp. 43–46, 2016.
- [52] A. Bajaj and O. P. Sangwan, “Study the impact of parameter settings and operators role for genetic algorithm based test case prioritization,” in *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM)*, 2019, pp. 1564–1569.
- [53] H. Bhasin, S. Gupta, and M. Kathur, “Regression testing using fuzzy logic,” *International Journal of Computer Science and Information Technologies*, vol. 4, no. 2, pp. 378–380.
- [54] S. Ghai and S. Kaur, “A hill-climbing approach for test case prioritization,” *International Journal of Software Engineering and Its Applications*, vol. 11, no. 3, pp. 13–20, Mar. 2017, doi: 10.14257/ijseia.2017.11.3.02.
- [55] E. K. Mece, K. Binjaku, and H. Paci, “The application of machine learning in test case prioritization - a review,” *European Journal of Electrical Engineering and Computer Science*, vol. 4, no. 1, Jan. 2020, doi: 10.24018/ejece.2020.4.1.128.
- [56] A. Kaur and D. Bhatt, “Hybrid particle swarm optimization for regression testing,” *International Journal on Computer Science and Engineering*, vol. 3, no. 5, 2011.
- [57] B. Suri and I. Mangal, “Analyzing test case selection using proposed hybrid technique based on BCO and genetic algorithm and a comparison with ACO,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 4, pp. 206–210, 2012.




- [58] J. K. Bajwa and R. Kaur, "An adaptive approach for test case prioritization in regression testing using improved genetic algorithm," *An International Journal of Engineering Sciences*, vol. 24, pp. 1–11, 2017.
- [59] T. Sachdeva, "Swarm intelligence techniques and genetic algorithms for test case prioritization," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 4, pp. 465–469, Apr. 2020, doi: 10.35940/ijeat.D6810.049420.
- [60] N. Sethi, S. Rani, and P. Singh, "Ants optimization for minimal test case selection and prioritization as to reduce the cost of regression testing," *International Journal of Computer Applications*, vol. 100, no. 17, pp. 48–54, 2014.
- [61] A. Ansari, A. Khan, A. Khan, and K. Mukadam, "Optimized regression test using test case prioritization," *Procedia Computer Science*, vol. 79, pp. 152–160, 2016, doi: 10.1016/j.procs.2016.03.020.
- [62] E. Ashraf, K. Mahmood, T. Ahmed, and S. Ahmed, "Value based PSO test case prioritization algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 1, 2017, doi: 10.14569/IJACSA.2017.080149.
- [63] A. Kaur and A. P. Agrawal, "A comparative study of Bat and Cuckoo search algorithm for regression test case selection," in *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*, Jan. 2017, pp. 164–170, doi: 10.1109/CONFLUENCE.2017.7943143.
- [64] S. F. Ahmad, D. K. Singh, and P. Suman, "Prioritization for regression testing using ant colony optimization based on test factors," in *Intelligent Communication, Control and Devices Advances in Intelligent Systems and Computin*, 2018, pp. 1353–1360.
- [65] P. Dhareula and A. Ganpati, "Cuckoo search algorithm for test case prioritization in regression testing," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 3, pp. 6004–6009, Sep. 2019, doi: 10.35940/ijrte.C4488.098319.
- [66] B. Manaswini and A. Rama Mohan Reddy, "A cat swarm optimization based test case prioritization technique to perform regression testing," *International Journal Of Recent Technology And Engineering*, vol. 8, no. 1, pp. 2677–2682, 2019.
- [67] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed, and M. D. M. Suffian, "Test case prioritization using firefly algorithm for software testing," *IEEE Access*, vol. 7, pp. 132360–132373, 2019, doi: 10.1109/ACCESS.2019.2940620.
- [68] R. Vats and A. Kumar, "Test case prioritization using cat swarm optimization," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 5, pp. 8142–8148, Oct. 2020, doi: 10.30534/ijatcse/2020/175952020.
- [69] N. Reda, A. Hamdy, and E. A. Rashed, "Multi-objective adapted binary bat for test suite reduction," *Intelligent Automation & Soft Computing*, vol. 31, no. 2, pp. 781–797, 2022, doi: 10.32604/iasc.2022.019669.
- [70] U. Sharma and V. Rastogi, "Test case selection and prioritization for regression testing using fault severity," *International Journal of Computers & Technology*, vol. 2, no. 3, pp. 1171–1177, Jun. 2012, doi: 10.24297/ijct.v9i3.6814.
- [71] A. Pravin and S. Srinivasan, "Effective test case selection and prioritization in regression testing," *Journal of Computer Science*, vol. 9, no. 5, pp. 654–659, May 2013, doi: 10.3844/jcsp.2013.654.659.
- [72] R. Beena and S. Sarala, "Code coverage based test case selection and prioritization," *International Journal of Software Engineering & Applications*, vol. 4, no. 6, pp. 39–49, Nov. 2013, doi: 10.5121/ijsea.2013.4604.
- [73] O. Dahiya, K. Solanki, and S. Dalal, "Comparative analysis of regression test case prioritization techniques," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 4, pp. 1521–1531, Aug. 2019, doi: 10.30534/ijatcse/2019/74842019.

BIOGRAPHIES OF AUTHORS



Shahbaa I. Khaleel    was born in Mosul, Nineveh, Iraq, she received the B.S., M.Sc. and Ph.D. degrees in computer science from Mosul University, in 1994, 2000 and 2006, respectively, assistant prof. since 2011, and finally, Prof. degree on 2021. From 2000 to 2020, she was taught computer science, software engineering and techniques in the College of Computer Sciences and Mathematics, University of Mosul. She has research in a field computer science, software engineering, intelligent technologies. She can be contacted at shahbaaibrkh@uomosul.edu.iq.



Raghda Anan    was born in Mosul, Nineveh, Iraq. She received the B. S. degree in computer science from Mosul University, in 2012. She is a master student in the college of Computer sciences and mathematics, University of Mosul. She can be contacted at raghda.20csp5@student.uomosul.edu.iq.