IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 30, 2020, accepted September 15, 2020, date of publication September 25, 2020, date of current version October 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3026911

Test Suit Generation for Object Oriented Programs: A Hybrid Firefly and Differential Evolution Approach

MADHUMITA PANDA[1], SUJATA DASH[1], (Member, IEEE), ANAND NAYYAR[2,5], (Senior Member, IEEE), MUHAMMAD BILAL[3], (Member, IEEE), AND RAJA MAJID MEHMOOD[4], (Member, IEEE)

[1]Master of Computer Application (MCA), North Orissa University, Baripada 757003, India
[2]Graduate School, Duy Tan University, Da Nang 550000, Vietnam
[3]Department of Computer and Electronics Systems Engineering, Hankuk University of Foreign Studies, Yongin 17035, South Korea
[4]Information and Communication Technology Department, School of Electrical and Computer Engineering, Xiamen University Malaysia, Sepang 43900, Malaysia
[5]Faculty of Information Technology, Duy Tan University, Da Nang 550000, Vietnam

Corresponding author: Raja Majid Mehmood (rmeex07@ieee.org)

This work was partially supported by the Xiamen University Malaysia Research Fund (XMUMRF) under Grant XMUMRF/2019-C3/IECE/0007.

ABSTRACT In model-based testing, the test suites are derived from design models of system specification documents instead of actual program codes to reduce cost and time of testing. In search-based software testing approach, the nature inspired meta-heuristic search algorithms are used for automating and optimizing the test suite generation process of software testing. This paper proposes a concrete model-based testing framework; using UML behavioral state chart model along with the hybrid version of the two most popular nature inspired algorithms, Firefly algorithm (FA) and Differential Algorithm (DE). The hybrid algorithm is adopted to generate optimized test suits for the benchmark triangle classification problem. Experimental results evidently show that the hybrid FA-DE search algorithm outperforms the individual model-based Firefly and Differential Evolution algorithm's performances in terms of time complexity, better exploration and exploitation as well as variations in test case generation process. The framework generates optimized test data for complete transition path coverage of the available feasible paths of the example problem.

INDEX TERMS Firefly algorithm, differential evolution, hybrid FA-DE algorithm, object oriented testing, path coverage, search-based testing, model-based testing.

I. INTRODUCTION

The software development organizations spend more than two third of the project development cost on product testing. The main intention of testing is to define some specific set of test suites that are capable enough to reveal the hidden errors/mistakes associated with the software under test thus avoiding bugs or system failures in future [37], [53]. The two most universally adapted testing strategies followed by testers are functional testing commonly known as black box testing and structural testing, popularly known as white box testing [67]–[71]. White box testing tests the logical flows, the key control flow paths, and program logics of the software under test. The black box testing tests the functions or modules of the software, verifying the outputs generated for a given set of inputs.

Currently in major sectors like banking, stock markets, telecommunication, health management, university management, internet applications, rocket launching systems and mobile applications etc., almost in every domain the widespread use of object-oriented programming approach is noticed. The popularity of the object-oriented programming concept is due to its modular structure and specific features like encapsulation, polymorphism, inheritance, dynamic binding etc. [22] that make the development and modification of applications quite simple in comparison to structured programming style. The object-oriented testing paradigm, popularly known as grey box testing was introduced in late 80s, the main challenges and complexities encountered in this approach is the testing of the specific

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

features introduced by object-oriented programming concepts. The specific programming features that make the object-oriented programming popular also made it too critical to test. Therefore, a different testing style known as model-based testing approach was adapted for object-oriented testing [39]. In the model-based testing approach the test cases are derived directly from system specification and design documents, i.e. mainly from the dynamic models also known as behavioural models of software systems instead of actual programme codes [49], it is a very tough task and an open research problem [22], [27]. The existing testing strategies are still incompetent to generate optimal set of test cases or test suites for critical path coverage and therefore the researcher community is still trying to figure out some new frameworks or methodologies for the complete automation of the process of object-oriented testing [30]. In last two decades researchers have started applying the meta-heuristic search algorithms [40], [43]–[45] to the field of model-based testing [22], [31], [34], [39], [78] for generating optimized test cases or test sequences.

Nature inspired algorithms are gradually being hybridised [86], [87] keeping in mind the best features of different algorithms, [1], [18], [24], [32], [48], [63] to obtain more variation and quality in the solutions, as some metaheuristics are very efficient in exploration where as others are good at exploitation. The hybrid metaheuristic techniques are created by combining two search algorithms where balance between exploration and exploitation is maintained, one algorithm is better in exploration and other one in exploitation [86]. The Firefly Algorithm (FA) is a stochastic, population-based metaheuristic, that has proved its efficiency in solving NP hard problems, in various fields of engineering and industry like wireless network design, market pricing, structural optimization, robotics etc. [11], [16]. Many versions of Firefly algorithms are applied for solving various problems in the fields like cryptanalysis [18], graph colouring [17], speech reorganization [3], for improving the speed of convergence, for diagnosing Parkinson's disease, for feature selection [6]–[8] and to provide microarray data for cancer prediction [1], [3], [7]. Similarly, the Differential Evolution (DE) algorithm is based on the evolutionary principle of survival of the fittest; it's a very popular algorithm, proving its excellence in global optimization, at the same time hybridization of DE with several other algorithms have provided excellent results [29].

Keeping the above research findings in mind, in this paper a novel FA-DE algorithm has been proposed along with a framework for model-based testing of object-oriented programs, using UML behavioural state chart model. The proposed hybrid algorithm provides good exploitation feature using Firefly Algorithm (FA) and enhanced exploration feature using Differential Evolution (DE) algorithm to generate balanced test suits for the benchmark triangle classification problem. Initially, the UML State chart models are converted to state chart graphs (SCG), then feasible test sequences are extracted from the SCG graph and finally model-based hybrid FA-DE algorithm is applied to select a suitable set of test suites from a vast set of possible test suits for testing those feasible paths. Efficiency of the algorithm is verified using the benchmark triangle classification problem. Hence, the objectives of the proposed proposal would be achieved by implementing the following modules.

1) Development and generation of object-oriented test suites for triangle classification problem using FA based model.
2) Development of a DE based object-oriented model for generation and verification of test suits for the same classical problem.
3) Development of FA-DE hybrid model for test suit generation and its performance verification using the triangle classification problem.
4) An extensive set of experiments has been performed on a well-known benchmark triangle classification problem and a comparative analysis with state-of-the-art methods including single and hybrid metaheuristics.

The remaining parts of the paper are arranged as follows. Section II provides a detailed literature review, section III provides explanations of the hybrid metaheuristic algorithms with parameter settings; Section IV describes the proposed framework for test suit generation, and Section V explains the detailed experimental set up and statistical analysis of the experimental results. Finally, the paper provides the conclusions and possible future directions in Section VI.

II. LITERATURE REVIEW

At present the newly emerging subdomain of testing, search-based software testing had shown promising results in the field of software testing, where metaheuristic nature inspired algorithms, predominantly evolutionary Genetic algorithms (GA), genetic programming (GP), bio-inspired algorithms including particle swarm optimization (PSO), Ant colony optimization (ACO), Firefly Algorithm (FA), and Cuckoo search (CS) algorithms were employed in automating the process of test case generation and test case prioritization [33], [68], [76]. The nature inspired evolutionary algorithms mainly the Genetic algorithms or genetic programming and swarm based metaheuristic algorithms [19], [32] including PSO, ACO, CS and FA were used for generating test data or test cases targeting specific coverage criteria [12].

In last two decades as there was a paradigm shift from structured programming approach towards object, oriented programming approach a new testing methodology known as model based testing was adapted to test the object-oriented programs. Then researchers started applying the metaheuristic search algorithms both evolutionary as well as swarm based [43]–[45] to the field of model-based testing [22], [31], [34], [39], [70] for generating optimized test cases or test sequences.

The metaheuristic algorithms, also known as nature-inspired algorithms are robust optimizers coming under the category of stochastic algorithms. The metaheuristic algorithms were developed replicating the natural processes such

as the gravitational force of attraction, various laws of physics, harmonics, principles of chemistry, the adaptability features followed by the nature i.e. the process of evolution and the intelligentsia shown by its various species starting from micro-organisms like bacteria to birds, honey bees, flies, fishes, frogs, monkeys, wolfs and many more [81].

The popular and widely applied metaheuristic algorithms include Genetic Algorithms (GAs) [33], [73], [75], Particle Swarm Optimization algorithm (PSO) [41], [42], [59], [60], [65], Artificial Bee Colony algorithm (ABC) [25], Cuckoo Search algorithm (CS) [15], [19], [26], [35], Firefly Algorithm (FA) [11], [20], [32], [37], [46], Differential evolution (DE) algorithms [25], [29], [30], [85], Ant colony optimization(ACO) [55] etc.

Though more than 300 types of nature inspired metaheuristic algorithms are available in the literature [82], the most widely accepted and popular primary algorithms mainly include the Genetic algorithms(GAs), Differential evolution algorithms(DE), Artificial Bee colony algorithms(ABC), particle swarm optimization algorithms(PSO), fireflies algorithm(FA) and ant colony optimization algorithm(ACO). The metaheuristic bio-inspired algorithms have proved their proficiency in solving complex real world application problems in versatile fields of Engineering like language recognition using firefly algorithm [3], in speech recognition where the parameters of a fuzzy neural network are optimized using Firefly algorithm [13] in electronics circuit design, in problems of traffic optimization using popular metaheuristics like GA, DE, ACO, GP(genetic programming), ABC etc [36], in enhancing the image contrast using FA with chaotic sequence and data classification using ACO [8], [52], in healthcare the firefly model is used for Parkinson's disease diagnosis and classification [5]–[7], in Robotics, where swarm based Glow worm optimization algorithm with multimodal functions was used for collective robotics applications similarly GA and PSO Algorithms were used for Intelligent Robot Path Optimization [16], [59].

Gradually with wider use of metaheuristics it was observed that these algorithms are not suitable for solving all kinds of problems and a specific metaheuristic algorithm shows excellent performance in solving a particular problem whereas the same algorithm shows worst performance in solving another type of problem, therefore one or a few metaheuristic algorithms cannot be standardized to get optimized solutions for all types of problems. Problem definition, fitness function, and parameters play a major role in the performance of those metaheuristic algorithms. Therefore gradually the hybrid approaches combining two metaheuristics are adapted for solving complex problems, a hybrid model based TVIW-PSO-GSA algorithm and SVM was applied for Classification Problems [60], a binary hybrid grey wolf optimization technique was used for feature selection [61] and some added applications like layout designing, graphics and art designing [28].

The model-based testing framework using metaheuristics seems too complex and difficult, as a small number of standard papers are available and the area still remains challenging to the researchers, Kari and Kumar [15] used a model based Cuckoo search algorithm for test suite optimization, similarly Shirole and Kumar [34] provided a detailed analysis on the model-based test case generation processes using UML behavioural models, Utting *et al.* [39] provided a thorough taxonomy of the various model based testing approaches. Since last two decades more than seventy percent of research work in the field of model-based testing using metaheuristics, is mainly based on genetic algorithms [33], [72] or genetic programming [9]. Many researchers have already suggested that instead of individual metaheuristic algorithms the hybrid models are more suitable for providing better-optimized solutions due to their combined exploration and exploitation capabilities [86]. In literature, very less number of papers [84] are available, where hybrid algorithms are applied for model-based testing of object-oriented programs. In this section out of the vast number of available papers, the basic papers have been provided, which have proved to be useful in providing a detailed analysis and through understanding of the metaheuristic algorithms, hybrid metaheuristic algorithms, application of those metaheuristic and hybrid metaheuristic algorithms in the broad area of search based testing and model based testing.

Uzun *et al.* [77] have described that search based software testing is the emerging field of software engineering domain where the software testing problem is reformulated as a search problem to select an appropriate and specific set of test suites using some metaheuristic algorithms and the fitness function is defined on the basis of certain coverage metric. They have mentioned that although a number of search-based optimization techniques are available but still a very little theoretical analysis is available regarding the suitability of the problem for specific testing problems, they also suggested that a hybrid global search approach may be suitable for solving the test data generation problem.

Daniel *et al.* [87] have made an exhaustive literature study of bio-inspired algorithms during past few decades and they find out that the number of metaheuristic bio-inspired optimization approaches have reached to such unprecedented levels that it may dark the future prospects of this research field. They have addressed the problem by proposing two comprehensive, principle-based taxonomies thus allowing the future researchers to organize the existing and upcoming algorithmic developments on the basis of two well defined criteria i.e. the source of inspiration and the behavior of each algorithm. In this work they have reviewed more than three hundred publications dealing with nature-inspired and bio-inspired algorithms, and the most interestingly they revealed that more than one-third of the reviewed bio-inspired solvers are versions of classical algorithms mainly Genetic algorithms, particle swarm optimization algorithm, Ant colony optimization algorithm etc.. The authors have also suggested that the hybridisation of the existing algorithms may produce new algorithmic behaviours and the results may be able to solve complex problems which still remain unsolved using

existing single metaheuristic algorithms and once solid proof is provided that the hybrid approaches are able to compensate the increasing complexity then those approaches would be gradually incorporated using existing taxonomies of metaheuristic algorithms.

Khari and Kumar [15] have presented a cost effective and time efficient Cuckoo Search (CS) algorithm for test data optimization; they have provided a detailed statistical study for validating their results.

Zhang *et al.* [48] have proposed a hybrid firefly algorithm combining the advantages of firefly (FA) and differential evolution (DE) algorithms and also verified the performance of the hybrid algorithm using benchmark unimodal and multimodal functions. The experimental results show that the hybrid firefly algorithm had better performance than the original versions of FA, DE and PSO algorithms in convergence rate and in avoidance of getting trapped in local minima.

Nayyar *et al.* [51]–[53], Durbhaka *et al.* [54], Nayyar and Nguyen [55], Diwaker *et al.* [56], and Gheisari *et al.* [57] have provided a detailed understanding on evolutionary algorithms including GA, and swarm based algorithms including ACO, ABO, PSO, Glow worm, Cockroach swarm optimization, Cat swarm optimization, Dolphin echo location, Eagle strategy, monkey search algorithm etc., highlighting the various computational models, the versatile approaches along with their applications in the newly emerging complex fields of engineering including IOT, AI, Big data, Data mining and Robotics.

Panda and Dash [23] have provided a detailed overview of the popular metaheuristic algorithms since last two decades including Cuckoo Search(CS), Gravitational search algorithm(GSA), Genetic Algorithms(GA), Particle swarm optimization(PSO), Differential Evolution(DE) and Artificial Bee Colony algorithm(ABC) and have compared the performances of the algorithms to generate test data for path coverage based testing.

Sahoo *et al.* [84] have used a hybrid bee colony algorithm combining Particle swarm optimization(PSO) and Bee Colony (ABC) algorithms along with unified modelling language (UML) combination diagram for optimized test data generation using the UML state chart diagram and sequence diagram of an ATM system.

Panthi and Mohapatra [27] used firefly algorithm for generating optimized and prioritized test sequences from UML state machine diagrams. The firefly algorithms have proved themselves in solving numerical optimization problems, i.e. the NP hard problems and also the firefly algorithm reduces the overall computational effort by 86 and 74 percent respectively in comparison to Genetic algorithm(GA) and particle swarm optimization algorithm(PSO).

Srivastava *et al.* [37] have used a modified firefly algorithm along with guidance matrix for generating optimal test paths. They claim that their methodology is capable of generating optimal discrete independent test paths that are highly useful in software testing.

Guohua *et al.* [83] have presented a very recent detailed review on popular strategies known as ensemble strategies that could be incorporated to different stages of population-based algorithms, to enhance their efficiency, precision and robustness. These ensemble techniques improve the computational intensiveness of the population-based algorithms by providing versatile tools and paradigms to design a better algorithm that would be able to handle versatile optimization problems. Here the adapted controlling parameters are updated automatically depending on the type and complexity of optimization algorithms, i.e. the algorithm tuning is performed by those automatically adapted parameters instead of normal hit and trial method.

Grosan and Abraham [86] stated that although evolutionary computation has solved many practical problems in engineering, business, commerce, etc., still sometimes they fail to give better performance due to poor parameters selection or in appropriate problem representation. This is in accordance with the No Free Lunch theorem, which states that for any algorithm, any high performance over one class of problems is paid by poor performance in another class, therefore the need for hybrid evolutionary algorithms is emphasized and they explained the several possibilities for hybridization of the metaheuristic algorithms along with they presented a detailed review of the available hybrid frameworks using PSO, ACO, Bacteria foraging algorithm(BFO) and some generic hybrid evolutionary architectures developed during the last couple of decades.

The object-oriented testing paradigm was introduced in late 80s, where the main challenge was the testing of the specific features of the object-oriented programming concepts such as inheritance, multiple inheritance, polymorphism, encapsulation and overloading [22]. At the same time, it is also unavoidable to ensure the quality and dependability of the software. The best way out for reducing cost and handling program complexity during software testing of object-oriented software is the complete automation and optimization of its entire product testing phase.

In earlier 90s Panda and Dash [22], tried to apply the traditional testing techniques in testing object-oriented programs and recommended that these programs can only be tested by considering the massage passing between the objects or change of states of the objects i.e. taking into consideration the dynamic behaviour of the system. Therefore, generating test cases from design documents rather than codes would be more appropriate.

Sharma *et al.* [33] proposed that the model-based testing approach mainly concentrates on test case generation and test result evaluation using a model. A software model mainly describes the system behaviour in terms of the input sequences accepted by the system, a set of conditions, actions, the data flow between its modules and routines.

Panda and Dash [22] described that the code-based testing approaches are not applicable in object-oriented programs testing, due to the specific features of Object-Oriented programming concepts like data encapsulation, data abstraction,

dynamic binding etc. and therefore model-based testing approaches are used for the testing of object oriented programs. Some of the popular software testing models include the unified modelling language (UML) models, finite state machines (FSM) models, Markov chains model and formal models. The design artefacts, mainly the UML behavioural models are mostly used by researchers for testing; these include Use case model,

Activity model, State chart model, sequence model, object model and component model. A group of researchers also used combinatorial models fusing two UML models like Use case and sequence model, Activity and sequence model, state chart and sequence model etc.

Ananya and Swapan [2] demonstrated that the UML models cannot be used directly for testing; some intermediate representation is required for using those UML models in testing. A lot of research work is conducted in this direction and the popular techniques include symbolic execution, OCL(Object control language), Directed graph(DG), System sequence diagram(SSD), Sequence diagram graph(SDG), Extended control flow graph(ECFG), State machine graph, Activity graph, Message flow graph(MFG), Use case diagram graph(UDG),Communication tree, object oriented graph(OOG).

Srivastava *et al.* [37] revealed that it is a very tough task to analyse UML models, in particular the behavioural models as they capture the dynamic system behaviour. Specifically, for object-oriented programs, many researchers propose the automation of software testing process but till date test sequence generation and complete test coverage remains an open research problem. The existing testing strategies cannot guarantee the exact and optimal set of test cases or test suites for coverage of the critical paths as well as quality of testing.

Saeed *et al.* [49] explained that optimization algorithms are broadly classified into two primary categories; first, one is the deterministic algorithms and the second one is stochastic algorithms. Deterministic algorithms include the algorithms like Hill Climbing, Newton-Raphson Method, Simplex method etc., for similar set of initial values; these algorithms obtain similar set of final values. The stochastic algorithms always produce a new set of solutions even though they begin with the same set of initial points. These algorithms include some advantages as well as disadvantages. The advantages include shared information, preservation of good solutions and very few chances of getting confused with local best as the global best. The disadvantages include, these are complex metaheuristic algorithms, require a lot of parameter settings and show better performance with large data sets.

In model-based testing approach appropriate test suites can be extracted from UML models to test object-oriented programs, a detailed view of the literature available on model-based testing in last two decades is presented in TABLE I. Though around more than two decades of research work has been carried out, still we require some concrete framework to automatically generate optimized and prioritized test suites for the hassle free model-based testing of object-oriented programs and software.

In the literature, few papers are available for model-based test data generation employing firefly algorithm [27], [37]. The authors have used the ATM state chart model and vending machine model as a case study for their problem. Srivastava *et al.* [37] have used a combined graph reduction technique with firefly algorithm to generate discrete and independent paths. Panthi and Mohapatra [27] have generated optimized and prioritized test sequences from UML state machine models, specifically generated test sequences for the composite states. Samuel *et al.* [31] generated test cases from UML state machine diagrams by applying transformed predicate functions.

The exhaustive search of the available literatures of last five years showed the availability of only one research work based on model-based testing using hybrid Bee colony algorithm [84]. In this work optimised path sequences are generated from UML combinational diagrams, the state-chart sequence diagram system graph (SCSEDG) of the ATM withdrawal operation. The hybrid Bee colony algorithm is developed by merging PSO and Bee colony algorithm where first the initial population was randomly generated and the fitness function of individual solutions was calculated and then the candidate solutions were ranked according to the fitness value. Afterwards the solutions were divided into two groups, the best solution are kept and the worst solutions are replaced with a copy of the best solutions, then the two metaheuristic algorithms are separately applied to get best optimal solutions. Here the solutions are path sequences and the optimized path is only one, the best path having minimum cost.

This work has much similarity with our work in terms of the model-based testing approach using UML diagrams as well as hybrid algorithms, but this approach cannot be compared with our work, as our objective is to generate test data for every feasible path targeting transition path coverage, also in our case the case study is the benchmark triangle classification problem having four paths and in the above work the case study is for ATM withdrawal operation, and their objective is to select only one path sequence having minimum cost.

After performing an in-depth study of the available literature on model-based testing using metaheuristic algorithms, it was observed that very few papers are available on the use of hybrid metaheuristic in the field of model based testing; no concrete framework is still available for automatic test suite generation with complete path coverage. In order to overcome the above difficulties, this paper proposed a novel FA-DE algorithm which generates optimal test suits fulfilling complete transition path coverage for model-based testing of object-oriented programs.

TABLE 1. Literature review on model-based testing.

Year of Publication	Title of paper	Author name	What Research work is proposed from abstract	Advantages/ disadvantages
2018	Model-Driven Architecture Based Testing: A Systematic Literature Review	Uzunet al. [77]	Model driven architecture-based testing mainly derives test cases from the system architecture models. Here the authors have analyzed 739 papers and identified 31 of them exactly related to the area and provided a thorough review identifying the current research directions and the methodologies followed.	They have suggested that the field is still novice and the approaches followed in those papers are quite different from each other in the test goals, modeling approaches as well as in the test data generation process.
2015	Review of model-based approach for automating the test case generation for Object Oriented Systems.	Rajvir et al.[78]	The authors have provided a detailed survey on the publications related to the various test data generation techniques using UML behavioral models.	This paper suggested that the most difficult part of the model-based testing approach is to decide and fit a specific model to a specific problem and no concrete guidelines are available to measure the suitability of the techniques used.
2013	UML Behavioral Model-based Test Case Generation: A Survey	S.Mahesh [79]	This paper provided a survey on the different UML model-based testing techniques, giving special emphasis on the UML behavioral diagrams like sequence, state chart and activity diagrams.	This work provided a clear classification of the various research approaches used for test case generation, like formal specification, Model-based graph based, UML specification etc. It suggested that the different behavioral models are able to capture the different features of the same scenario and therefore appropriate test cases can be designed using this model-based testing approach.
2011	Automated test cases Generation for object Oriented software	Shanthi et al.[80]	This paper proposed an automatic test case generation technique using UML class models, Genetic algorithm, and Depth first search algorithm along with transition path coverage criteria.	The specification-based test case generation approach is generating test cases from UML class models. But it is not validated with any other test case generation approach.
2009	Object-Oriented Software and UML-Based Testing: A Survey Report	S. Supavita.[81]	This report summarized noteworthy published works in object-oriented software testing and UML-based testing areas over the past 15 years. It included innovations in the concepts, techniques, and approaches.	It suggested some areas which could be explored further in the domain of UML model-based object-oriented testing, like application of symbolic execution techniques, data flow testing techniques and aspect-oriented software.
2006	A Taxonomy of Model-based Testing	Utting et al. [39]	This paper provided an overall review on the model-based testing paradigm in seven dimensions. It provided the advantages disadvantages along with the approaches used and issues arise in model-based testing process.	This research work provided a concrete conceptual framework to analyze and categorize different model-based testing approaches and their usability along with the supporting tools available. It also provided the research challenges and limitations encountered in this field of model-based testing and test data generation.

TABLE 1. *(Continued.)* **Literature review on model-based testing.**

2002	Model-based Testing of Object-Oriented Systems	Rumpe et al.[82]	It argued that an approach using models as central development artifact needs to be added to the portfolio of software engineering techniques, to further increase efficiency and flexibility of the development as well as quality and reusability of results. Then test case modeling is examined in depth and related to an evolutionary approach to model transformation. A number of test patterns are proposed that have proven helpful to the design of testable object-oriented systems. In contrast to other approaches, this approach uses explicit models for test cases instead of trying to derive (many) test cases from a single model.	The proposal made in this paper is part of a pragmatic approach to model-based software development. This approach uses models as primary artifact for requirements and design documentation, code generation and test case development and includes a transformational technique to model evolution for efficient adaptation of the system to changing requirements and technology, to optimize architectural design and fix bugs. To ensure the quality of such an evolving system, intensive sets of test cases are an important prerequisite. They are modeled in the same language, namely UML, and thus exhibit a good integration and allow us to model system and tests in parallel. Therefore, we can conclude that techniques such as model-based development, model evolution and test-first design will change software engineering and add new elements to its portfolio.

III. PROPOSED HYBRID MODEL-BASEDTESTING APPROACH

Nature inspired algorithms are good at solving many complex problems in different fields of science and engineering efficiently. There are several stochastic model-based nature inspired algorithms and out of them the Firefly (FA) and Differential Evolution (DE), have outperformed in various complex optimization problems. The detailed descriptions of FA and DE along with their algorithms have been presented in the Algorithm 1 and Algorithm 2. However, both algorithms have some inherent limitations such as FA searches nearby local regions and take more time to converge, whereas DE explores more randomly due to its mutation operator and gets premature convergence. Therefore, integrating the respective merits of FA and DE, a hybrid algorithm, denoted as FA-DE is proposed in Algorithm 3 to obtain quality test suites for testing object-oriented programs.

A. FIREFLY ALGORITHM

Firefly algorithm (FA) is a population-based optimization technique in the swarm intelligence family and it is proposed by Yang and He [46], Yang *et al.* [47]. This algorithm is inspired by the swarming and flashing light characteristics of the fireflies. In the summer night group of fireflies in the sky produce flashing light for two fundamental reasons, to attract their partners for mating and to protect themselves from potential predators [8].

However, the flashing lights follow two physical laws: first, the light intensity (I) is inversely proportional to the distance (r) in the form of Iα1/r2 light intensity deceases as the distance increases and second, the intensity of light exponentially decreases due to absorption of light in the air.

Algorithm 1 Firefly Algorithm (FA)

Input: Objective function f(x), number of decision variables (D), parameters bounds [L, U]
Output: Optimal candidate solution

1. Initialize parameters: Population size (N), γ, β_0, α
2. Initialize a set of random fireflies x_i, $\{1, \cdots, N\}$
3. Compute light intensity f (x_i) for all fireflies
4. Define the light absorption coefficient (γ)
5. while $t<MaXGen$ do
6. for $i \leftarrow 1$ to N do
7. for $j \leftarrow 1$ to N do
8. Compute the distance r_{ij} between two fire flies x_i and x_j using Euclidean distance in Eq.(4)
9. if light intensity f (x_i) $< f$ (x_j) then Less-brighter firefly moves towards more-brighter firefly
10. Compute attractiveness varies with absorption parameter (γ) and distance (r_{ij}) using Eq.(3)
11. Move firefly x_i towards firefly x_j using Eq.(5)
12. Update new solution $x_i(new)$ and its light intensity f (x_i) (new)
13. end if
14. end for
15. end for
16. Sort light intensity of the fireflies to update the best solution
17. $t = t + 1$
18. **end while**

Therefore, in FA algorithm the intensity of light is associated with fitness value of the cost function to be optimized

[11]. The FA can be formulated based on following three rules:

1) All fireflies assumed to be unisex so any firefly can be attracted to other fireflies irrespective their sex for mating.
2) The attractiveness is proportional to light intensity of the fireflies.
3) The light intensity of a firefly is determined by the cost function that is to be optimized.

From the above rules, it is understood that FA has been designed using two important issues: i) the variation of light intensity and ii) formulation of the attractiveness that is inversely proportional to the distance. Hence, the attractiveness of a firefly can be established with its flashing light intensity which in turn is considered as fitness value of the corresponding firefly. In addition, flashing light of firefly also gets absorbed in the air. Hence, the light intensity (I) inversely varies with distance (r) and adsorption (γ) which can be derived as follows:

$$\boldsymbol{I = I_0 e^{-\gamma r}} \tag{1}$$

where $\boldsymbol{I_0}$ denotes the light intensity of the firefly with distance $\mathbf{r = 0}$, and the light absorption is assumed with a fixed light absorption coefficient. In order to avoid singularity at $\mathbf{r = 0}$ in Eq. (1) we can combine inverse square law and adsorption using Gaussian form as follows.

$$\boldsymbol{I = I_0 e^{-\gamma r^2}} \tag{2}$$

Similarly, the attractiveness (β) is also the function of distance and adsorption. The attractiveness of a firefly is determined based on light intensity of the fireflies in its neighborhood in Gaussian form and it is defined as follows.

$$\boldsymbol{\beta = \beta_0 e^{-\gamma r^2}} \tag{3}$$

where β_0 is the initial attractiveness of a firefly which is initialized with a constant value at distance $r = 0$. The terms light intensity I and attractiveness β are by some means equivalent. The term light intensity indicates total light emitted by a firefly, the term attractiveness denotes amount of light that someone can see and being observed by other fireflies at a distance (r). The distance between any two fireflies in a group $\mathbf{x_i}$ and $\mathbf{x_j}$ can be computed by Euclidean distance in the Cartesian space, as follows.

$$\boldsymbol{r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{k=d} \left(x_{i,k} - x_{j,k} \right)^2}} \tag{4}$$

where d denotes the dimensionality of each firefly; x_i and $\mathbf{x_j}$ are the $\mathbf{i^{th}}$ and $\mathbf{j^{th}}$ fireflies of the population. In the FA, movement of each firefly takes place based on the principle that the $\mathbf{i^{th}}$ firefly attracts another $\mathbf{j^{th}}$ firefly when $\mathbf{j^{th}}$ firefly is more attractive than $\mathbf{i^{th}}$ firefly. The movement of the fireflies is formulated in the algorithm as follows

$$\boldsymbol{x_{i+1} = x_i + \beta_0 exp^{-\gamma r_{ij}^2} \left\| x_{ik} - x_{jk} \right\| + \alpha \left(rand - \frac{1}{2} \right)} \tag{5}$$

where α is a randomization parameter and third term generate random number in the range [−1, 1] from the Gaussian distribution. Eq. (5) represents new position of the i^{th} firefly consists of three terms: the current position of i^{th} firefly, move to j^{th} firefly which is more attractive, and a random walk in the range of [−1, 1]. Finally, the steps of Firefly optimization are summarized in Algorithm 1.

B. DIFFERENTIAL EVOLUTION ALGORITHM

Differential Evolution (DE) is a stochastic population-based optimization technique in the evolutionary algorithms family and it is proposed by Rainer Storn and Kenneth Price [29]. DE algorithm works on the principle of evolution theory of nature i.e. survival of the fittest [39]. It primarily consists of two operators namely, mutation and recombination. In DE, the main role is played by the mutation operator and it is followed by the recombination operator. In evolutionary algorithms, each candidate solution is known as a genome or chromosome. However, in DE candidate solutions are referred to as vectors with three different names such as target vectors, donor vectors, and trial vectors. The classical DE algorithm comprises four major steps: initialization, mutation, recombination, and selection. Initialization step randomly initializes population and controlling parameters of the algorithm for one time. Then, last three steps of the algorithm are repeated iteratively till maximum number of generations is reached or termination criterion is satisfied. The candidate solutions are referred as target vectors that are randomly generated within the search space restricted by lower and upper bounds such as,

$$\boldsymbol{x_{min} = \left(x_{min,1}, x_{min,2}, \ldots, x_{min,d} \right) \ and}$$
$$\boldsymbol{x_{max} = \left(x_{max,1}, x_{max,2}, \ldots, x_{max,d} \right).}$$

Then, we can initialize the j^{th} component of the i^{th} target vector as follows, where rand is a uniformly distributed random number that varies within 0 and 1.

$$\boldsymbol{x_{i,j} = x_{min,j} + rand^* \left(x_{max,j} - x_{min,j} \right)} \tag{6}$$

1) MUTATION

After the initialization step, DE algorithm generates i^{th} donor/mutant vector v_i^n corresponding to each target/parent vector x_i^n in n^{th} iteration using mutation operator. Two most popular mutation operators are formulated as mentioned below:

$$\boldsymbol{DE/rand/1 : v_i^n = x_{R1}^n + F * \left(x_{R2}^n - x_{R3}^n \right)} \tag{7a}$$
$$\boldsymbol{DE/best/1 : v_i^n = x_{best}^n + F * \left(x_{R1}^n - x_{R2}^n \right)} \tag{7b}$$

where random numbers $R_1/ = R_2/ = R_3$ within the range of $[1, 2, \ldots.., N_p]$; One of the control parameter F is known as scaling factor which is a positive real number in the range of [0, 2]. In this work, we use these two mutation operators in the experiment that offer better result over other variants of mutation strategies. Unlike GA, in the DE algorithm the target vector is not involved in mutation operation.

Algorithm 2 Differential Evolution Algorithm

Input: Fitness function f , number of decision variables (D), parameters bounds (lower bound (lb), upper bound (ub))
Output: Best candidate solution
1: Initialize control parameters: maximum number of generations ($MaXGen$), population size (Np), Scaling factor (F) and crossover probability (P_c)
2: Initialize random candidate solutions $\mathbf{x_i}$
3: **while** t<$MaXGen$ **do**
4: **for** $i \leftarrow 1$ to Np **do**
5: Select randomly three target vectors $\mathbf{x_{R_1}}$, $\mathbf{x_{R_2}}$, and $\mathbf{x_{R_3}} \in Np$
6: Generate the donor vector $\mathbf{v_i}$ using mutation operation
7: **end for**
8: **for** $i \leftarrow 1$ to Np **do**Perform binomial crossover operation
9: Generate a set of N random integer indices (K) $\in \{i, \cdots, D\}$
10: Generate a set of N random real numbers (r_j) $\in U$ (0, 1)
11: **end for**
12: **for** $i \leftarrow 1$ to Np **do**
13: **for** $j \leftarrow 1$ to D **do**
14: **if** $r_j \leq Pc$ OR $j = K_i$ **then**
15: $\mathbf{u_{i,j}} = \mathbf{v_{i,j}}$
16: **else** if $r_j > Pc$ and $j \neq K_i$
17: $\mathrm{u_{i,j}} = \mathrm{x_{i,j}}$
18: **end if**
19: **end for**
20: **end for**
21: **for** $i \leftarrow 1$ to Np **do**
22: Bound trail vector $\mathbf{u_i}$ within sample space
23: Evaluate fitness f ($\mathbf{u_i}$)
24: Perform greedy selection between f ($\mathbf{u_i}$) and $f(\mathbf{x_i})$ and update target vector $\mathbf{x_i}$
25: **end for**
26: $t = t + 1$
27: **end while**

2) CROSSOVER

In order to increase the diversity in the search space, next recombination (crossover) operator combines the components of donor/mutant vector v_i^n with the target vector x_i^n to obtain trial/offspring vector, $\boldsymbol{u_i^n} = \left(\boldsymbol{u_1^n}, \boldsymbol{u_2^n}, \cdots, \boldsymbol{u_d^n}\right)$. In this strategy, the trail vector is directly involved for the crossover operation. The DE algorithm basically uses two crossover operators such as binomial (uniform) and exponential (two-point modulo). Binomial crossover is applied on a number of D components of the donor vector based on uniformly generated random numbers which vary from 0 and 1and is less than or equal to a pre-defined control parameter called crossover rate (Pc).

$$\boldsymbol{u_{i,j}^n} = \begin{cases} \boldsymbol{v_{i,j}^n}, & \boldsymbol{if\ rand_{i,j} \leq P_c\ or\ j = k} \\ \boldsymbol{x_{i,j}^n}, & \boldsymbol{if\ rand_{i,j} > P_c\ and\ j \neq k} \end{cases} \tag{8}$$

The binomial crossover is defined as in equation (8), Where k is a randomly generated natural number in the range $\{1, 2, \cdots, \mathrm{d}\}$, $\mathrm{rand_{i,j}}$ is a randomly generated real number that varies in between [0, 1]. Next, the exponential crossover, we first need to choose a random integer number (n) between $\{1, 2, \cdots, \mathrm{d}\}$. Before nth component, all components of target vector are copied to the trail vector, than nth variable from donor vector is directly copied to corresponding position of the trial vector. For subsequent components, real valued random numbers are generated between [0, 1]. If $\mathrm{rand_{i,j}} \leq \mathrm{Pc}$ Then copy the components from donor vector to trial vector. When $\mathrm{rand_{i,j}} > Pc$, copy remaining components of target vector to the trial vector. In this work, both the crossover operators have been used to maintain diversity in the solutions; here the solutions are the different test cases. From the experiment results, it was revealed that exponential crossover produces variation in the test cases, which is needed for all paths coverage.

3) SELECTION

Based on fitness values of target (parent) and trial (offspring) vectors, the greedy selection scheme is adopted to decide survival condition of the vectors to the next generation ($n + 1$). The selection procedure is expressed as:

$$\boldsymbol{x_i^{n+1}} = \begin{cases} \boldsymbol{u_i^n}, & \boldsymbol{iff}\ \left(\left(\boldsymbol{u_i^n}\right) \geq \left(\boldsymbol{x_i^n}\right)\right) \\ \boldsymbol{x_i^n}, & \boldsymbol{otherwise} \end{cases} \tag{9}$$

where f(.) is the objective (cost) function to be maximized. It is necessary to mention that we consider only maximization problem in this paper.

C. PROPOSED HYBRID FIREFLY-DIFFERENTIAL EVOLUTION ALGORITHM

In order to improve the quality of the solutions and overcome the limitations of both individual Firefly and Differential evolution algorithms, this paper introduced a novel hybrid model-based framework established on the hybrid version of Firefly(FA) and Differential evolution(DE) algorithms. In proposed work, quality of the solutions means to generate test cases for the transition path coverage of the feasible paths present in the objective function, from all parts in the search space. It is only possible when both exploitation and exploration features are simultaneously justified in a metaheuristic optimization technique. Each metaheuristic algorithm has its own capacity to improve either exploitation or exploration characteristics.

Recently, researchers are showing interest to use hybrid approach, [86] by combining more than one algorithm together in a single framework, to improve quality of the solutions, by giving importance to both exploitation and exploration features. In this work, firefly and differential evolution algorithms are combined to form an efficient hybrid metaheuristic approach, referred to as hybrid FA-DE algorithm, in which the algorithms FA and DE will work together for simultaneously increasing the exploitation and exploration capability.

As a result, the gap between exploitation and exploration decreases, that increases the number of test cases for each path. The FA has good exploitation capability because each firefly moves towards another brighter one neighbour firefly. In this way, all the fireflies of the population make different clusters based on their attractiveness property. Similarly, DE is an evolutionary algorithm which has higher exploration capability due to its mutation operator that is used for all candidate solutions in the population to increase randomness in the search space.

In FA, every firefly gets attracted to another brighter firefly and if the firefly is unable to find any neighbourhood brighter one then it tries to find another firefly through a random walk [20], [21], [47]. Here the DE algorithm [48] replaces the random walk feature used for exploration of the search space for the desired firefly. Differential evolution algorithm uses mutation and crossover operators only on those fireflies that cannot find a brighter one. When any firefly is unable to find a nearest brighter one then it is considered that the firefly may be the local best. At this point, in this case the DE operators such as mutation and crossover promote the firefly in avoiding the situation of getting trapped in local minima and provide quality test suits for object-oriented programs. The detailed steps of the hybrid FA-DE is given in Algorithm 3.The hybrid FA and DE algorithm has been designed as follows, FIGURE.1.

An individual firefly is attracted to its neighborhood firefly when the later firefly is brighter in the population based on its fitness. This process continues until the maximum number of fireflies in the population is reached. To ensure diversity, only some specific fireflies are selected for the execution of DE with greater random number [0, 1] in comparison to the rate of selection of FA. Thus it can be concluded that the DE improves exploration capability and convergence speed. Therefore, the combination of FA and DE algorithms is useful to increase the random search behavior of the proposed hybrid algorithm, for getting high quality solutions in the form of uniform number of test cases for all paths coverage.

D. TIME COMPLEXITY OF HYBRID FIREFLY AND DIFFERENTIAL EVOLUTION ALGORITHM

All of the metaheuristic algorithms are used to solve NP hard complex engineering optimization problems in polynomial time. They are less complex in design and easy to implement. Hence, they produce efficient results compared to other conventional optimization techniques.

Nowadays, many hybrid metaheuristic algorithms have been proposed for engineering design applications to provide consistent and better results than the individual metaheuristic algorithms. In practice, the design of hybrid model-based algorithm is to eliminate the limitations of individual metaheuristic algorithms at the cost of a little bit more time complexity. In this work, a hybrid FA-DE algorithm is proposed by combining the best exploitation property of FA and exploration feature of DE, with the expectation of generating consistent test suits by covering entire problem space in less

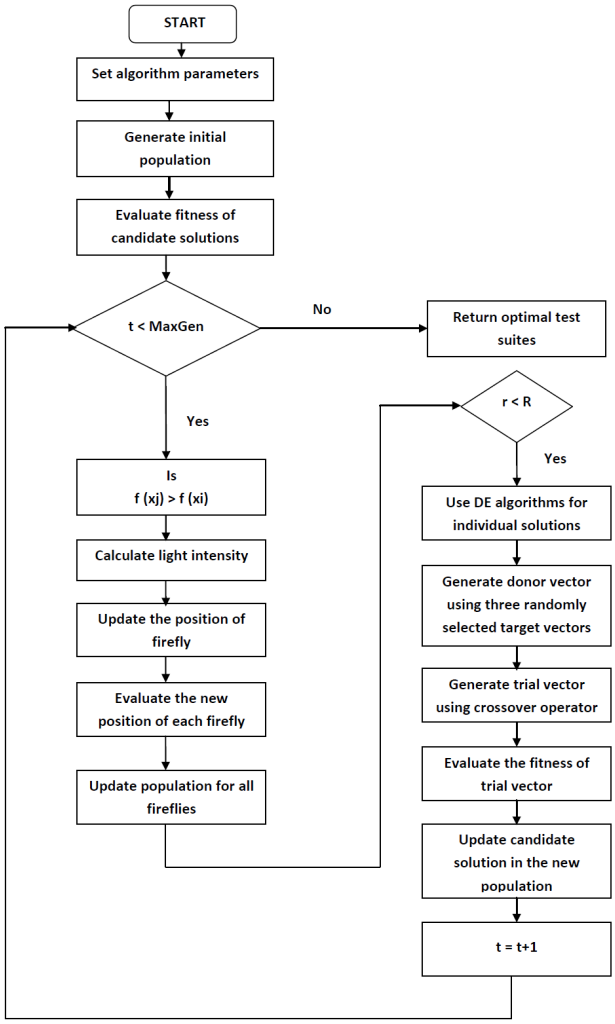

FIGURE 1. Proposed hybrid FA-DE flowchart.

computing time. Firefly algorithm consists of two inner loops of maximum size (n), the population size and one outer loop of size t, which is the maximum number of generations of the FA. Hence the time complexity of Firefly in worst case is $O(n^2t)$. Similarly, DE has the same time complexity which is $O(n^2t)$. In case of small n (our case, $n = 50$ maximum) and small t (our case, $t = 30$ maximum), the computation cost is relatively linear with respect to t. In general, the major computational time is expended in the evaluations of fitness function, which is same for all metaheuristic algorithms. In case of the proposed hybrid FA-DE algorithm, if a firefly fails to get neighbourhood brighter firefly then DE algorithm is executed in place of the random walk feature in FA. Hence, the proposed hybrid FA-DE algorithm takes same constant time i.e., $O(n^2t)$ which is relatively inexpensive than the standard FA, DE and other hybrid algorithms as mentioned in the literature [46].

IV. PROPOSED FRAMEWORK

This paper suggests a framework as shown in FIGURE. 2 to generate a set of suitable test suits from UML state chart

Algorithm 3 Proposed Hybrid FA-DE Algorithm

Input: Objective function f , number of decision variables (D), parameters bounds [L, U]

Output: Optimal test suite

1: Initialize parameters: Population size (N), γ, β_0, α, F and Pc

2: Initialize random candidate solutions $\mathbf{x_i}$, $\{1,\cdots,N\}$
//Initial position of each firefly

3: Evaluate fitness of each firefly f ($\mathbf{x_i}$)
//Measure light intensity of each firefly

4:**while** $t<MaXGen$ **do** //
MaXGen: Maximum number of generations

5: **for** each firefly $\mathbf{x_i} \in \{1, \cdots, N\}$ **do**

6: **for** each firefly $\mathbf{x_j} \in \{1, \cdots, N\}$ **do**

7: **if** f ($\mathbf{x_j}$) > $f(\mathbf{x_i})$ **then**

8: Calculate the Euclidean distance between $\mathbf{x_i}$ and x_j using Eq.(4)

9: Calculate attractiveness (β) using Eq.(3)

10: Update $\mathbf{x_i}$ to new solution $\mathbf{x_i}$ (*new*) by using Eq.(5)

11: Evaluate the objective function based on fireflies f ($\mathbf{x_{new}}$)

12: Update new population based on fitness

13: **end if**

14: **end for**

15: **end for**

16: **for** each target vector $\mathbf{x_i} \in \{1, \cdots, N\}$ **do**

17: **if** *rand* () < R **then** Execute DE algorithm when random value [0, 1] <Selection rate(R)

18: Select three target vectors randomly $\mathbf{x_{R1}}$, $\mathbf{x_{R2}}$, and $\mathbf{x_{R3}} \in N$

19: Generate donor vector for each target vector $\mathbf{v_i}$ using mutation operator Eq.(7)

20: Generate trial vector $\mathbf{u_i}$ using crossover operator as follows Eq.(8)

21: Evaluate the objective function based on trial vector f ($\mathbf{u_i}$)

22: Update individual candidate solution of the new population based on fitness

23: **end if**

24: **end for**

25: $t = t + 1$

26: **end while**

27: Return optimal test suit

model using hybrid FA-DE algorithm. Initially the UML state chart model is converted to state chart graph. Then the start node of the state chart graph(SCG)is assigned weight=1, edges are also assigned with weight=1, here the parent node number is the weight for the child node and if a child node has many parent nodes then the sum of all parent node weights is the weight of the child node [19]. Then Depth first search is applied to traverse the SCG graph, tracing the feasible paths and the total path cost, i.e. the sum of node weights and edge weights assigned to each path. Here the total path weight is the fitness function for each feasible path [24]. After calculation of the fitness function, the fireflies are

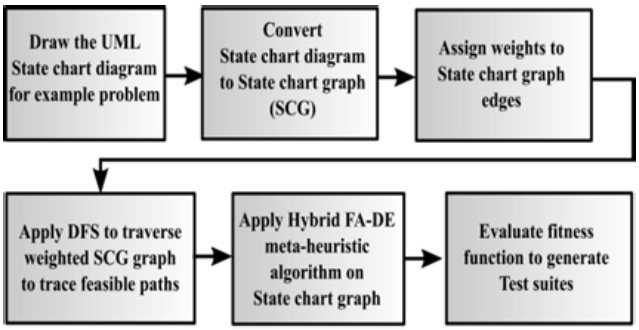

FIGURE 2. Proposed framework for test suites generation.

generated randomly at each node and the sum of the edge and node weights is the fitness function of respective fireflies. Finally, the hybrid FA-DE metaheuristic algorithm is applied to generate test suits for path based coverage of the feasible paths.

V. EXPERIMENTS AND RESULT ANALYSIS

Here in the example problem, the UML state chart is developed using ArgoUML, and then the test suites are generated using MATLAB R2016b. The experiments were performed on Intel Core TM i3 CPU, 2.0GHz speed, 4GB RAM, running on 64-bit windows.

This paper suggested a framework to generate optimized test suits for the model-based testing of object-oriented programs using UML state chart model and hybrid metaheuristic FA-DE algorithm. In this framework for test suite generation, first of all the UML state chart model is developed for the example case study, 'the benchmark triangle classification problem', using ArgoUML case tool as shown in FIGURE 3. Then the state model is converted to start chart graph (SCG), as shown in FIGURE 4, where the states of the state chart model are considered as nodes of the SCG graph. In the next step DFS graph traversal algorithm is applied to traverse the SCG graph for generating the feasible path sequences and from those feasible path sequences the total path weight of respective path sequences is calculated. The path weight is the fitness function of the problem, and it's a maximizing optimization problem.

The hybrid FA-DE optimization algorithm has been used to generate test suites and here the path-based coverage criteria determines the fitness of the candidate solutions. In this work, the performance of the proposed hybrid FA-DE algorithm was implemented and compared among metaheuristic algorithms including FA, DE, PSO, CS and our recently published hybrid particle swarm optimization and gravitational search algorithm(PSO-GSA) [22] and hybrid Cuckoo search and simulated annealing(CS-SA) [24] metaheuristic algorithms for test suites generation. The first experiment determines optimal control parameter values of the proposed algorithm, then the second experiment compares four metaheuristic algorithms to point out the importance of the proposed hybrid algorithm, and finally the third experiment compares the

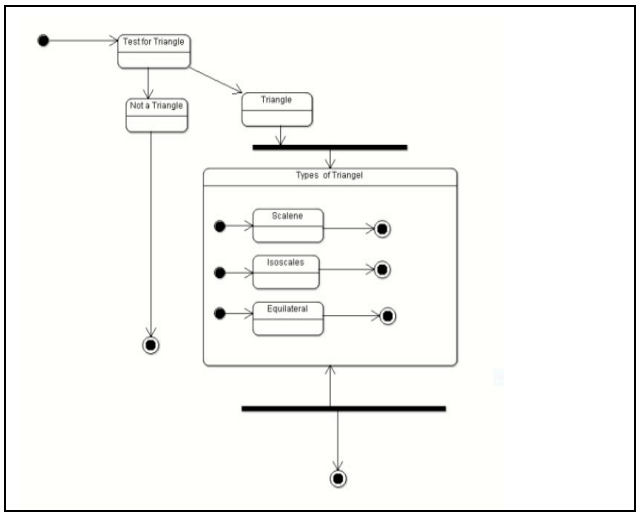

FIGURE 3. State chart model for triangle classification problem.

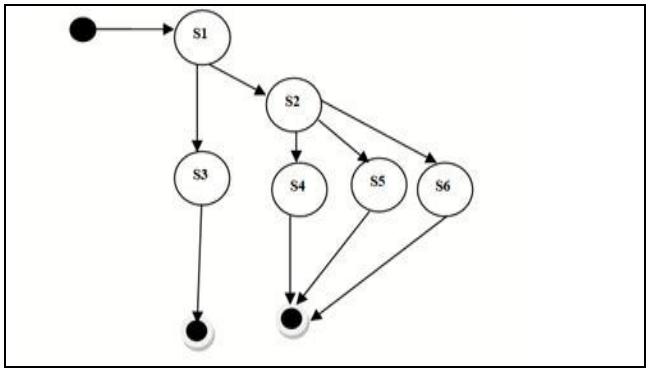

FIGURE 4. State chart graph (SCG) triangle classification problem.

effectiveness of the proposed FA-DE algorithm with FA, DE, PSO, CS, PSO-GSA, CS-SA algorithms.

A. EXPERIMENTAL SETUP

The Classic Triangle classification problem is the bench mark problem used by many researchers, [22], [23] since last three decades in the domain of software testing specifically test data generation [34]. The generated test suites must hold test data that satisfies the triangle characteristics and distinctive attributes of this problem, thus putting it in a very specific group of benchmark software testing problem. Separate sets of data are required for different categories of triangles like isosceles, scalene and equilateral, which is too complex to figure out manually [25]. The triangle classification problem has been used as a case study for generating test suites using metaheuristic algorithms like Cuckoo search, Particle swarm optimization etc., [22]–[26], and is already recommended as a bench mark problem in the domain of test cases and test data generation. Therefore, the triangle classification problem is used in this work as a case study to generate test suites using metaheuristic FA, DE and hybrid FA-DE algorithms.

TABLE 2. Path sequences for triangle classification problem.

Path	Path Sequence	Type
Path 1	S1-S3	Not-a-Triangle
Path 2	S1-S2-S4	Scalene
Path 3	S1-S2-S5	Isosceles
Path 4	S1-S2-S6	Equilateral

The UML state chart model of the example triangle classification problem is shown in FIGURE 3 and its state chart graph (SCG) is depicted in FIGURE 4. As shown in FIGURE 4, the triangle classification problem has six states, and four feasible paths. The state S1 is the test for triangle, S3 is not for a triangle, and S2 is a composite state showing a triangle, S4 state is for scalene, S5 state is for isosceles, and S6 is for equilateral. The TABLE II clearly includes the name of the paths, state sequences for the respective paths and the conditions that the respective path sequences satisfy.

B. EXPERIMENT 1: PARAMETER TUNING OF THE PROPOSED ALGORITHM

The performance of the metaheuristic algorithms is very much sensitive to their control parameters values. In this work, different combination of parameters values have been tried on the proposed FA-DE algorithm to get test suite for all the feasible paths of the benchmark program. The two common control parameters of all metaheuristic algorithms are the maximum number of generations and population size. In this work all the metaheuristic algorithms were examined with three different sizes of populations and generations i.e. 10, 20, and 30. First of all keeping number of generations constant, the model-based algorithm was executed repeatedly for the above three population sizes with different combinations of control parameters. For FA-DE algorithm five control parameters were needed to be tuned, out of which FA contains three control parameters: mutation coefficient(α), attraction coefficient (β_0) and light absorption coefficient(γ)and DE comprises of the other two control parameters: scaling factor (F) and crossover rate (P_C). The range of values for $\alpha = (0.1, 0.2, 0.3, 0.4, 0.5)$, $\beta 0 = (1, 2, 3)$, $\gamma = (1, 2, 3)$, $F = (1.1, 1.2, 1.3, 1.4, 1.5)$, and $P_C = (0.6, 0.7, 0.8, 0.9, 1)$ obtained 10, 125 different configurations of control parameters. The proposed FA-DE algorithm was run 30 times for each configuration of the control parameters to tune the optimal control parameters values. The optimal control parameter values of the proposed FA-DE algorithm are given in TABLE III.

C. EXPERIMENT 2: NUMERICALEXPERIMENTS AND COMPARISIONS

In order to analyse the behaviour of each metaheuristic algorithm in a large search space, all the experiments were

TABLE 3. Proposed hybrid FA-DE parameters.

Parameters	Description	Values
	Firefly Algorithm (FA)	
β_0	Initial value attraction coefficient	2
γ	Light absorption coefficient	2
α	Mutation coefficient	0.3
alpha_damp	Mutation coefficient damping ratio	0.97
	Differential Evolution (DE)	
R	Selection rate for DE algorithm	0.4
F	Scaling factor for mutation	1.4
P_C	Crossover rate	0.7

performed for the search space bounded with the range [−10,000 to 10,000]. In addition, the experiments were conducted keeping population size fixed at 30 with varying the total number of generations to 10, 20, and 30. Similarly the experiments were repeated keeping the number of generation fixed at 10 with varying population size to 10, 20, and 50. For each configuration of parameters population and generation, each algorithm was run 30 times to compute descriptive statistics on the test suite results. The descriptive statistics computed for the result of test data includes minimum, maximum, mean, standard deviation to evaluate the effectiveness and suitability of the proposed FA-DE hybrid algorithm over individual FA and DE algorithms. The statistical results produced by individual FA, DE, and hybrid FA-DE algorithms were presented in TABLE IV and TABLE V. The test data generation at each generation is depicted in graphical manner for varying population size and numbers of generations are represented in Figure 5 and Figure 6. In the TABLE IV and TABLE V, the minimum test cases interpret minimum path coverage i.e., the minimum number of test cases generated for a particular path in one of the algorithm executions. It points out the lower bound in test cases generation for a particular path. Similarly, the maximum test cases infer maximum path coverage i.e., the maximum number of test cases generated for a particular path in one of the algorithm executions. It points out the upper bound in test cases generation for a particular path.

In this experiment, if a minimum value is zero, then it indicates that in at least one of the executions that particular path has not been covered. If a maximum value is zero that means that no test case has been generated for a path in 30 times execution of the algorithm. In the results, it is clearly observed that one minimum value is zero in the proposed FA-DE hybrid algorithm. The mean test cases indicate average number of test cases generated for a particular path in the total number of executions of the algorithm. The proposed FA-DE algorithm ensured large mean value of the test cases for all paths; however, it is not true for all paths in case of FA and DE. Another important statistic is the standard deviation that indicates the variations in test cases generation in the number of times of execution. Depending on the statistics in TABLE IV and TABLE V, mainly the mean and standard deviation values are closer for all paths in the proposed FA-DE than FA and DE.

In addition, it was also observed that FA has more exploitation capability that leads to some path coverage, a large number of times than other paths. Similarly, DE has better exploration capability that implies more paths were covered with closer values than FA. These results motivate us to hybrid FA and DE to produce a set of balanced and stable statistics values with much improved generation of test suites as compared to FA and DE. In many cases the FA and DE generated zero data for path 2 and 3 whereas the proposed hybrid FA-DE algorithm provides uniform number of test data for every path. By comparing TABLE IV, and TABLE V, it is clearly evident that for a small population and generation, FA and DE, are not able to generate adequate number of test suits for all the four paths. On the other hand, FA-DE is quite efficient in generating stable and uniform test suits for all the four paths specifically for path3 which is a critical path of the problem. The FIGURE 5 (a, b, c) and 6 (a, b, c) depict the tests suites generated for the triangle classification problem by DE, similarly,FIGURE 5 (d, e, f) and 6(d, e, f) show the test cases generated by FA and FIGURE 5 (g, h, i) and 6 (g, h, i) show the test cases generated by the proposed hybrid FA-DE algorithm. The above derived inference aligns with the projected graph depicted in FIGURE 5 (g, h, i) and FIGURE 6 (g, h, i). Hence, the proposed hybrid FA-DE algorithm generates optimal test suits uniformly for all the test paths which are pictorially shown using bar chart in FIGURE 7.

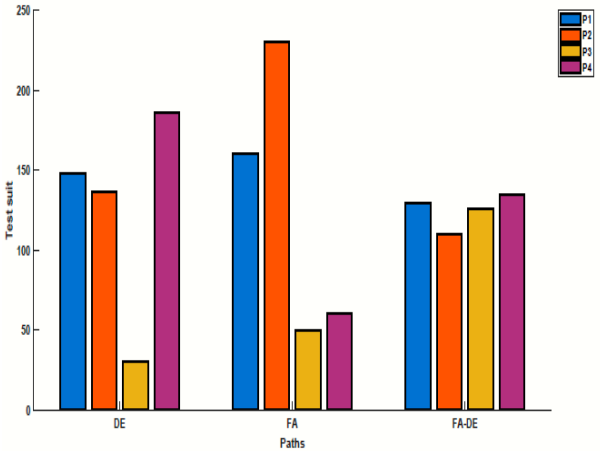

FIGURE 7. Comparison of test suits generation using DE, FA and FA-DE algorithms.

D. EXPERIMENT 3: STATISTICAL ANALYSIS OF METAHEURISTIC ALGORITHMS

In this experiment, the effectiveness of test suits generation by the proposed hybrid FA-DE was analyzed and compared with existing widely used metaheuristic algorithms such as PSO, DE, FA, and CS. These algorithms were mainly chosen keep-

TABLE 4. Descriptive statistics for the test suite generation using hybrid FA, DE, and FA-DE with fixed population size (30) and varation in the number of generations(10, 20, 30).

Generation (Gen)/Population (Pop)		Model-based Algorithm	Path1	Path2	Path3	Path4
Gen=10 Pop=30	Minimum	FA	50	2	0	139
		DE	44	117	0	8
		FA-DE	35	36	27	16
	Maximum	FA	152	17	12	246
		DE	134	229	44	48
		FA-DE	117	102	127	101
	Mean	FA	102.73	6.66	3.83	186.63
		DE	88.1	182.16	7.76	21.76
		FA-DE	74.2	75.16	79.4	71.23
	Standard Deviation	FA	25.36	3.82	3.36	27.36
		DE	21.06	24.87	7.86	9.05
		FA-DE	19.89	12.26	17.59	18.61
Gen=20 Pop=30	Minimum	FA	166	118	5	214
		DE	26	443	0	4
		FA-DE	11	13	39	127
	Maximum	FA	236	168	39	280
		DE	134	560	19	26
		FA-DE	269	263	296	281
	Mean	FA	192.46.	149.06.	20.5	237.96
		DE	51.9	530.26	5.7	12.03
		FA-DE	103.23	144.16	144.93	206.56
	Standard Deviation	FA	15.31	10.07	9.24	16.04
		DE	20.81	22.44	5.06	5.3
		FA-DE	62.57	47.54	54.67	40.23
Gen=30 Pop=30	Minimum	FA	194	159	8	312
		DE	38	789	0	4
		FA-DE	13	28	133	236
	Maximum	FA	314	249	68	461
		DE	88	859	13	19
		FA-DE	279	337	380	438
	Mean	FA	277.8	227.76	37.43	357
		DE	53.53	831.9	5.03	10.53
		FA-DE	111.96	197.53	256.26	334.23
	Standard Deviation	FA	26.32	18.34	15.06	39.55
		DE	12.34	15.36	3.99	4.35
		FA-DE	74.53	83.20	77.07	65.63

TABLE 5. Test suite generation using hybrid FA, DE, and FA-DE with fixed number of generation (10) and variation in population size (10, 20, 50).

Generation/ Population		Metaheuristic Optimization Algorithm	Path1	Path2	Path3	Path4
Gen=10 Pop=10	Minimum	FA	21	15	2	36
		DE	10	67	0	0
		FA-DE	3	4	14	29
	Maximum	FA	37	30	7	47
		DE	35	86	4	6
		FA-DE	31	33	40	50
	Mean	FA	30.1	24.2	4.1	41.6
		DE	18.3	77.8	1.5	2.7
		FA-DE	14.5	22.8	24.3	38.4
	Standard Deviation	FA	5.06	4.63	1.96	4.03
		DE	7.54	6.49	1.43	1.82
		FA-DE	9.59	9.46	7.95	9.44
Gen=10 Pop=20	Minimum	FA	55	28	1	69
		DE	11	132	0	0
		FA-DE	9	36	22	54
	Maximum	FA	82	54	11	103
		DE	54	180	7	10
		FA-DE	65	78	64	91
	Mean	FA	67.9	43.9	5.8	82.4
		DE	32.5	157.3	3.7	6.5
		FA-DE	37	51.9	35.2	75.9
	Standard Deviation	FA	7.85	7.85	3.11	10.46
		DE	13.07	14.05	2.71	3.40
		FA-DE	21.35	12.04	13.35	13.14
Gen=10 Pop=50	Minimum	FA	129	71	3	182
		DE	50	380	3	6
		FA-DE	12	86	40	113
	Maximum	FA	174	144	34	269
		DE	95	418	17	28
		FA-DE	187	221	227	219
	Mean	FA	159.7	117.7	17.3	208.3
		DE	76.1	398.8	10.2	14.9
		FA-DE	95.8	140.9	107.8	155.5
	Standard Deviation	FA	12.50	18.64	12.41	24.05
		DE	12.94	14.45	4.46	6.29
		FA-DE	59.18	47.61	65.37	35.77

ing in mind their popularity as well as successful in various fields of complex software engineering problems including test data generation, test sequence generation, test case prioritization, etc., At the same time, the proposed FA-DE was also compared with our published hybrid algorithms: PSO-GSA and CS-SA. The performance of these algorithms is

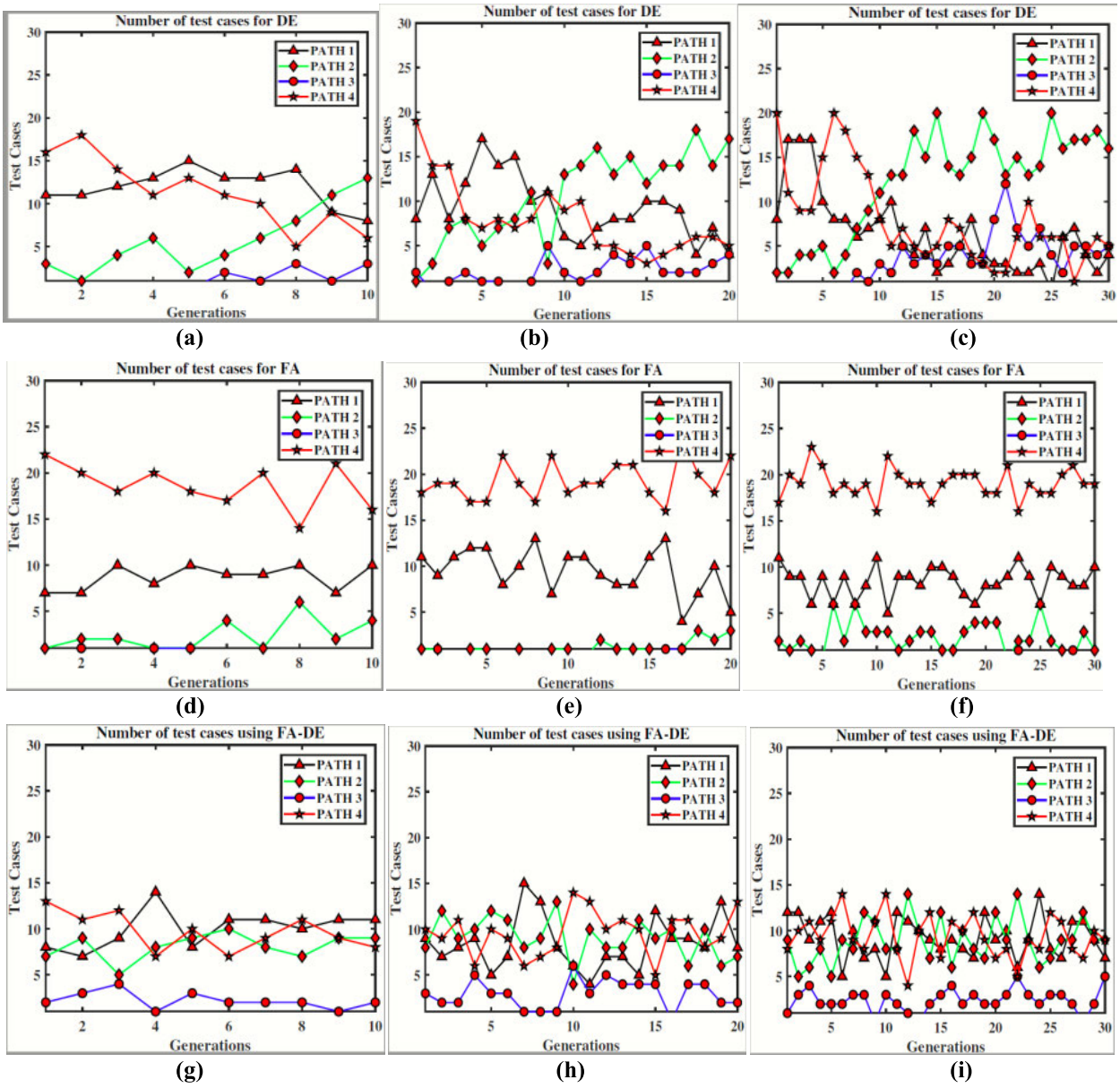

FIGURE 5. Test suite generation using DE (a, b, c), FA (d, e, f) and Hybrid FA-DE (g, h, i) with fixed population size 30 and variations in the number of generations (10, 20, 30).

compared using descriptive statistics and statistical hypothesis. In this experiment, number of test cases and execution time metrics were used for the statistical analysis and comparison. The descriptive statistics used to analysis test cases for different path of triangle classification problem and execution time of metaheuristic algorithms include: minimum, maximum, mean, median, mode, standard deviation, and standard error. In addition, to this single factor ANOVA is used to verify whether or not the results of the metaheuristic algorithms, those are significantly different from each other. Referring to the results in TABLE VI, it is clearly observed that mean, median; standard deviation values for all three paths such as Path1, Path2, and Path4 are nearly closer for DE and PSO algorithms which mean that both algorithms symmetrically cover three paths except Path 3. Based on the standard error it can be justified that DE is more stable than PSO. Furthermore, DE contains less control parameters than PSO that makes it quite easy for tuning. Similarly, by comparing statistical values, it was found that the FA is more stable and efficient than CS. Execution time in seconds of all mentioned algorithms for triangle classification problem were given in TABLE X. The statistical minimum, maximum, mean, median, and mode values of execution time for all the algorithms showed that PSO takes less time than all other algorithms. Based on the standard deviation and standard error values, PSO and DE take low values for all

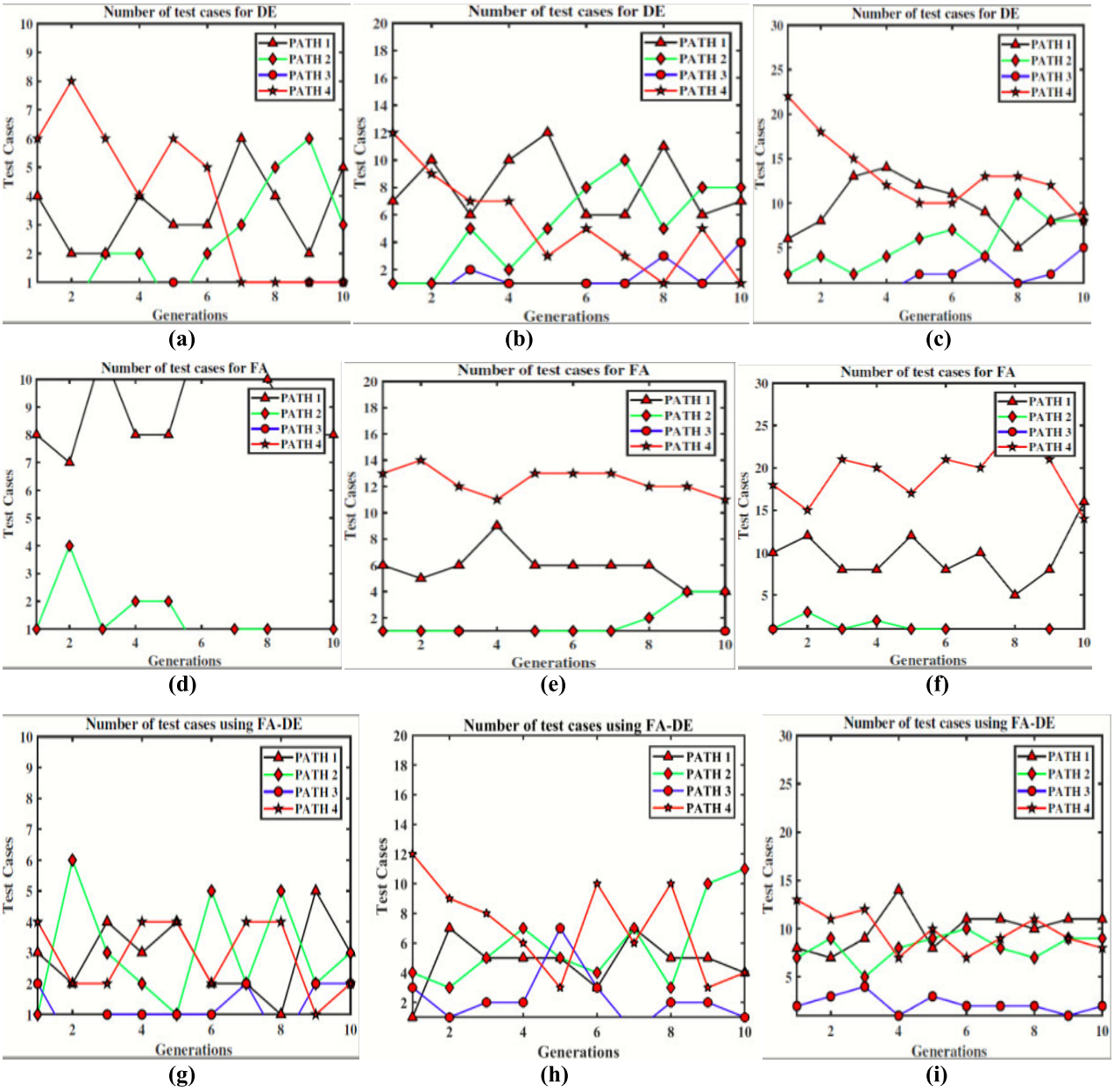

FIGURE 6. Test suite generation using DE (a, b, c), FA (d, e, f) and hybrid FA-DE (g, h, i) with fixed number of generation (10) and variations in the population size (10, 20, 50).

paths coverage in 30 times execution than other algorithms. Similarly, among FA and CS, FA takes less time than CS. Therefore DE and FA algorithms were selected for hybridization. Statistically, it was concluded that the performance of DE and FA is better than PSO and CS metaheuristic algorithms. Hence, it motivated us to hybrid FA-DE for better exploitation and exploration over a wide range of search space [−10000, 10000]. In order to compare the proposed FA-DE, our two recent works namely PSO-GSA and CS-SA were implemented to generate test suite.

The statistical results of test cases for four different paths were presented in TABLE VII. In which the proposed FA-DE algorithm was compared with PSO-GSA and CS-SA hybrid algorithms. The statistical results as mean and median values of FA-DE for all four paths are thinly closed and balanced than PSO-GSA and CS-AS closed and balanced than PSO-GSA and CS-AS algorithms. Similarly, variation of standard deviation and standard error of the three hybrid algorithms are very small with respect to individual metaheuristic algorithms as discussed in TABLE VI and the FA-DE outperforms PSO-GSA and CS-SA. Furthermore, the execution time of all algorithms discussed here is given in TABLE X. The statistical results are showing that the standard deviation and standard error of FA-DE is smaller than other two hybrid algorithms,

TABLE 6. Descriptive statistics for the test suite generation using FA, DE, PSO and CS with fixed population size (30) and number of generation (10).

Descriptive Statistics		Min	Max	Mean	Median	Mode	Standard deviation	Standard error
FA	Path 1	50	152	102.73	101.5	116	25.36	4.63
	Path 2	2	17	6.67	6	4	3.82	0.69
	Path 3	0	12	3.83	3	3	3.66	0.66
	Path 4	139	246	186.63	193	176	27.36	4.99
DE	Path 1	44	134	88.1	85.5	112	21.06	3.84
	Path 2	117	229	102.16	106	118	24.87	3.54
	Path 3	0	44	7.77	7	16	7.86	1.43
	Path 4	8	142	91.76	92	134	22.05	3.65
PSO	Path 1	41	166	94.7	95	123	27.55	5.03
	Path 2	53	229	138.7	136	151	43.55	7.95
	Path 3	0	75	26.53	14	5	26.18	4.78
	Path 4	12	102	36.73	29.5	28	22.01	4.01
CS	Path 1	15	36	24.56	25	25	5.66	1.03
	Path 2	26	278	260.93	269	265	44.68	8.15
	Path 3	0	6	3	3	2	1.43	0.26
	Path 4	1	10	3.5	3	3	1.92	0.35

i.e. PSO-GSA and CS-SA. In addition, through statistical hypothesis, it can be verified whether all the algorithms used in the experiments were performing in similar way or significantly different by using the test suites generated by the respective metaheuristic algorithms for all four paths along with their execution times. In order to check this hypothesis, the single factor ANOVA has been used in our experiment. Initially, a null hypothesis was specified, which states that Null hypothesis (H0): All algorithms used in this experiment have equally performed without significant difference. Then, one Alternative hypothesis was defined (H1): states that the set of algorithms used in the experiments are performance wise significantly different. Referring to the results of TABLE VIII and TABLE IX based on single factor ANOVA on test case generation and execution time of all algorithms, the Null hypothesis was rejected using F-critical value (2.14). The F test statistic (54.07) is greater than F-critical value (2.14) in TABLE VIII at 5 % level of significance. Hence, hypothesis (H0) is rejected and the result of test cases for all paths is significantly different among all algorithms. Also, another performance metric is execution time and the F test statistic (2703.15) was obtained using single factor ANOVA which is greater than F-critical value (2.14) at 5 % level of significance.

E. DISCUSSION

As the performance of the metaheuristic algorithms is affected by the improper selection of the control parameters, in the first experiment, the proposed algorithm was executed with different combination of control parameters to tune optimal control parameters.

In the second experiment, the performance of the proposed FA-DE was evaluated and statistically compared to FA and DE metaheuristic algorithms. The results presented in the TABLE IV and TABLE V revealed that the performance of the proposed FA-DE in large search space is superior over FA and DE. In the third experiment, using TABLE VI, TABLE VII and TABLE X showed the motivation behind the selection of the specific two algorithms, FA and DE for hybridization over other recently popular metaheuristic algorithms such as PSO and CS. Then, the performance of the hybrid FA-DE was statistically compared with our recently published PSO-GSA and CS-SA hybrid algorithms and it was well established that the proposed algorithm is superior in terms of generation of balanced test suites and inexpensive execution time.

Finally, using TABLE VIII and TABLE IX based on test cases for each paths and execution time of each algorithms TABLE X, by using single factor ANOVA, the null hypothesis was statistically rejected and alternative hypothesis was accepted which states that the test suite generated by different metaheuristic algorithms in the experiments are significantly different and by descriptive statistics it was showed that the proposed FA-DE hybrid algorithm can be an alternative method for test suite generation in object-oriented testing domain of software testing. From the analysis of experimental results and computation time, it can be concluded that the new hybrid FA-DE algorithm has the capability to achieve promising performance. The proposed hybrid FA-DE method has the following advantages: (1) the experimental results indicated that both algorithms indeed add complementary features results in increasing test suite coverage performance. (2) As DE algorithm is executed based on selection rate whenever, a firefly fails to get neighborhood brighter firefly then DE. Hence, the execution time of the proposed FA-DE algorithm is slightly higher than the execution time of FA algorithm. (3) Due to the balanced exploitation and exploration characteristics, the proposed method performs superior than existing metaheuristic algorithms. (4) Most of time, it generates test cases symmetrically for all critical paths. A comparison among all the algorithms is depicted pictorially in FIGURE 7. The major limitation of the proposed FA-DE algorithm is that the performance is sensitive to proper tuning of control parameters.

The proposed framework can be applied for testing real life problems, i.e. the test suites can be derived

TABLE 7. Descriptive statistics for the test suite generation using hybrid PSO-GSA, CS-SA and FA-DE with fixed number of generation (10) and population size (20).

Descriptive Statistics		Min	Max	Mean	Median	Mode	Standard deviation	Standard error
PSO-GSA	Path 1	56	79	65.93	67.5	71	5.91	1.08
	Path 2	11	124	107.16	110	105	19.52	3.56
	Path 3	35	69	53.9	52.5	49	9.67	1.76
	Path 4	53	90	69.66	68	59	9.85	1.79
CS-SA	Path 1	53	122	76.97	74	66	15.89	2.9
	Path 2	75	147	114.5	113.5	106	18.24	3.33
	Path 3	34	91	61.4	61	61	14.67	2.67
	Path 4	18	81	47.13	45.5	48	14.92	2.72
FA-DE	Path 1	35	117	74.2	74	88	19.89	3.63
	Path 2	36	102	75.16	76	76	12.26	2.23
	Path 3	27	127	79.4	82.5	83	17.59	3.21
	Path 4	16	101	71.23	74	72	18.61	3.39

TABLE 8. Single factor anova results based on test data generation of FA, DE, PSO, CS, PSO-GSA, CS-SA, and FA-DE algorithms.

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F- critical*
Between Groups	118751.4	6	19791.9	54.07	1.65E-39	2.14
Within Groups	74299.87	203	366.0092			
Total	193051.3	209				

TABLE 9. Single factor anova results based on execution time of FA, DE, PSO, CS, PSO-GSA, CS-SA, and FA-DE algoritms.

Source of Variation	SS	df	MS	F	P-value	F-critical
Between Groups	459.76	6	76.62	2703.15	1.1E-190	2.14
Within Groups	5.754	203	0.02			
Total	465.51	209				

TABLE 10. Execution time (in seconds) statistics of FA, DE, PSO, CS, PSO-GSA, CS-SA, and FA-DE algoritms.

Algorithms	FA	DE	PSO	CS	PSO-GSA	CS-SA	FA-DE
Minimum	0.036	0.051	0.041	0.057	0.039	4.079	0.06
Maximum	0.154	0.089	0.06	0.217	0.1	6.58	0.127
Mean	0.055	0.059	0.051	0.07	0.052	4.291	0.069
Median	0.046	0.057	0.051	0.061	0.042	4.199	0.095
Mode	0.044	0.052	0.051	0.06	0.04	4.199	0.095
Standard Deviation	0.02	0.006	0.004	0.029	0.018	0.443	0.01
Standard Error	0.0037	0.0016	0.0012	0.005	0.003	0.08	0.0018

for testing by following the proposed methodology without any further modifications or improvements. The software requirement specification document of any problem statement can be used for designing UML models and from those models test suits can be derived by replicating our proposed framework without further modifications.

VI. CONCLUSION

Automatic test suite generation in object-oriented programming is a challenging task. This paper proposes a novel hybrid FA-DE framework to generate optimized test suits targeting path-based coverage criteria of testing. The Firefly algorithm and Differential Evolution algorithm have proven their efficiency in solving many engineering optimization problems.

Our proposed framework has successfully generated optimal test suites for effective testing of object-oriented programs using UML state chart model.

The proposed framework is simulated using the benchmark triangle classification problem. The simulation results of the hybridized FA-DE algorithms clearly established the hybrid algorithms efficiency in uniform exploration and exploitation of the solution space, thus generating uniform test suits for all the feasible paths of the example problem, achieving full path coverage. Whereas the individual FA and DE algorithms, efficiently generated test suits for some specific paths. Therefore, the efficiency of the proposed Framework in generating uniform test suits for all the four paths of Triangle Classification problem can be attributed to the exploitation and exploration capabilities of the hybrid algorithm. This framework can be further enhanced by using the hybrid version of other metaheuristic algorithms and different UML models, taking into consideration large data sets. In addition to this the work will be extended by using ensemble strategies of metaheuristic algorithms [78] for test data generation in model-based testing of object-oriented programs.

REFERENCES

[1] A. Abdullah, S. Deris, M. S. Mohamadand S. Z. M. Hashim, "A new hybrid firefly algorithm for complex and nonlinear problem," *J. Distrib. Comput. Artif. Intell.*, vol. 151, no. 9, pp. 673–680, 2012.

[2] A. Kanjilal and S. Bhattacharya, "Static analysis of object oriented systems using extended control flow graph," in *Proc. IEEE Region 10 Conf. TENCON*, Nov. 2004, pp. 310–313.

[3] S. Biswas, S. Dash, and S. Acharya, "Firefly algorithm based multilingual named entity recognition for Indian languages," in *Proc. Int. Conf. Adv. Inform. Comput. Res.* Springer, 2018, pp. 540–552.

[4] A. Canuto, A. F. Neto, H. M. Silva, and J. C. Xavier-Júnior, "Population-based bio-inspired algorithms for cluster ensembles optimization," *Natural Comput.*, vol. 19, pp. 515–532, Mar. 2018.

[5] S. Dash and A. Abraham, "Kernel based chaotic firefly algorithm for diagnosing Parkinson's disease," in *Proc. Int. Conf. Hybrid Intell. Syst.* Shimla, India: Springer, 2018, pp. 176–188.

[6] S. Dash, R. Thulasiram, and P. Thulasiraman, "An enhanced chaos-based firefly model for Parkinson's disease diagnosis and classification," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, Dec. 2017, pp. 159–164.

[7] S. Dash, R. Thulasiram, and P. Thulasiraman, "Modified firefly algorithm with chaos theory for feature selection: A predictive model for medical data," *Int. J. Swarm Intell. Res. (IJSIR)*, vol. 10, no. 2, pp. 1–20, 2019.

[8] K. G. Dhal, M. Iqbal Quraishi, and S. Das, "Development of firefly algorithm via chaotic sequence and population diversity to enhance the image contrast," *Natural Comput.*, vol. 15, no. 2, pp. 307–318, Jun. 2016.

[9] M. K. Azam Atta-ur-Rahman, S. S. Sultan Dash, and N. Khan, "Automated testcase generation and prioritization using GA and FRBS," in *Computer and Information Science*, vol. 955. Singapore: Springer, 2019, pp. 571–584.

[10] J. M. Ferrández and R. Varela, "Bio-inspired population-based metaheuristics for problem solving," *Natural Comput.*, vol. 16, no. 2, pp. 187–188, Jun. 2017.

[11] I. Fister, I. Fister, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm Evol. Comput.*, vol. 13, pp. 34–46, Dec. 2013.

[12] M. Harman and B. F. Jones, "The SEMINAL workshop: Reformulating software engineering as a metaheuristic search problem," *ACM SIGSOFT Softw. Eng. Notes*, vol. 26, no. 6, pp. 62–66, 2001.

[13] T. Hassanzadeh, K. Faez, and G. Seyfi, "A speech recognition system based on structure equivalent fuzzy neural network trained by firefly algorithm," in *Proc. Int. Conf. Biomed. Eng. (ICoBE)*, Feb. 2012, pp. 63–67.

[14] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481–494, Oct. 2002.

[15] M. Khari, and P. Kumar, "An effective model-based cuckoo search algorithm for test suite optimization," *Informatica*, vol. 41, no. 3, pp. 363–377, 2017.

[16] K. N. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications," *Multiagent Grid Syst.*, vol. 2, no. 3, pp. 209–222, Sep. 2006.

[17] S. A. Lee, "K-phase oscillator synchronization for graph coloring," *Math. Comput. Sci.*, vol. 3, no. 1, pp. 61–72, Mar. 2010.

[18] J. Luthra and S. K. Pal, "A hybrid firefly algorithm using genetic operators for the cryptanalysis of a monoalphabetic substitution cipher," in *Proc. World Congr. Inf. Commun. Technol.*, Dec. 2011, pp. 202–206.

[19] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, pp. 1–17, Apr. 2017.

[20] H. C. Ong, S. L. Tilahun, W. S. Lee and J. M. T. Ngnotchouye, "Comparative study of prey predator algorithm and firefly algorithm," *Intell. Automat. Soft Comput.*, vol. 23, no. 1, pp. 1–8, Mar. 2017.

[21] S. K. Pal, C. S. Rai, and A. P. Singh, "Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems," *Int. J. Intell. Syst. Appl.*, vol. 4, no. 10, pp. 50–57, Sep. 2012.

[22] M. Panda and S. Dash, "Test-case generation for model-based testing of object-oriented programs," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.* Bhubaneswar, India: Springer, Jan. 2020, pp. 53–77.

[23] M. Panda and S. Dash, "Automatic test data generation using bio-inspired algorithms: A travelogue," in *Handbook of Research on Modelling, Analysis and Application of Nature-Inspired Metaheuristic Algorithms*. Hershey, PA, USA: IGI Global, 2018, pp. 140–159.

[24] M. Panda and S. Dash, "A framework for testing object oriented programs using hybrid nature inspired algorithms," in *Proc. Int. Conf. Adv. Inform. Comput. Res.* Singapore: Springer, 2018, pp. 531–539.

[25] M. Panda and P. P. Sarangi, "Performance analysis of test data generation for path coverage based testing using three meta-heuristic algorithms," *Int. J. Comput. Sci. Inform.*, vol. 3, no. 2, pp. 34–41, 2013.

[26] M. Panda, P. P. Sarangi and S. Dash, "Automatic test data generation using metaheuristic cuckoo search algorithm," *Int. J. Knowl. Discovery Bioinf. (IJKDB)*, vol. 5, no. 2, pp. 16–29, 2015.

[27] V. D. Panthi and P. Mohapatra, "Generating prioritized test sequences using firefly optimization technique," in *Computational Intelligence in Data Mining*, vol. 2. Bhubaneswar, India: Springer, 2015, pp. 627–635.

[28] F. K. H. Phoa, "A swarm intelligence based (SIB) method for optimization in designs of experiments," *Natural Comput.*, vol. 16, no. 4, pp. 597–605, Dec. 2017.

[29] K. V. Price, "Differential evolution," in *Handbook of Optimization*. Cham, Switzerland: Springer, 2013, pp. 187–214.

[30] O. Sahin and B. Akay, "Comparisons of Metaheuristic algorithms and fitness functions on software test data generation," *Appl. Soft Comput.*, vol. 49, pp. 1202–1214, Dec. 2016.

[31] P.Samual, R. Mall and A. K. Bothra, "Automatic test case generation using unified modeling language (UML) state diagrams," *IET Softw.*, vol. 2, no. 2, pp. 79–93, 2008.

[32] S. Sarbazfard and A. Jafarian, "A hybrid algorithm based on firefly algorithm and differential evolution for global optimization," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 6, pp. 95–106, 2016.

[33] C. Sharma, S. Sabharwal, and R. Sibal, "A survey on software testing techniques using genetic algorithm," 2014, *arXiv:1411.1154*. [Online]. Available: http://arxiv.org/abs/1411.1154

[34] M. Shirole and R. Kumar, "UML behavioral model based test case generation: A survey," *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 4, pp. 1–13, 2013.

[35] R. Soto, B. Crawford, R. Olivares, J. Barraza, I. Figueroa, F. Johnson, F. Paredes, and E. Olguín, "Solving the non-unicost set covering problem by using cuckoo search and black hole optimization," *Natural Comput.*, vol. 16, no. 2, pp. 213–229, Jun. 2017.

[36] S. Srivastava, and S. K. Sahana, "A survey on traffic optimization problem using biologically inspired techniques," *Natural Comput.*, vol. 19, pp. 1–15, Jan. 2019.

[37] P. R. Srivatsava, B. Mallikarjun, and X.-S. Yang, "Optimal test sequence generation using firefly algorithm," *Swarm Evol. Comput.*, vol. 8, pp. 44–53, Feb. 2013.

[38] Y. Tao and L. Zhang, "A multi-population evolution stratagy and its application in low area/power FSM synthesis," *Natural Comput.*, vol. 18, pp. 139–161, Nov. 2017, doi: 10.1007/s11047-017-9659-5.

[39] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Softw. Test., Verification Rel.*, vol. 22, no. 5, pp. 297–312, Aug. 2012.
[40] T. Vantuch, I. Zelinka, A. Adamatzky, and N. Marwan, "Perturbations and phase transitions in swarm optimization algorithms," *Natural Comput.*, vol. 18, pp. 579–591, May 2019.
[41] C.-F. Wang and K. Liu, "An improved particle swarm optimization algorithm based on comparative judgment," *Natural Comput.*, vol. 17, no. 3, pp. 641–661, Sep. 2018.
[42] H. Wang, L. L. Zuo, J. Liu, W. J. Yi, and B. Niu, "Ensemble particle swarm optimization and differential evolution with alternative mutation method," *Natural Comput.*, to be published, doi: 10.1007/s11047-018-9712-z.
[43] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*. London, U.K.: Luniver Press, 2010.
[44] X. S. Yang, "Metaheuristic optimization," *Scholarpedia*, vol. 6, no. 8, p. 11472, 2011.
[45] X. S. Yang, S. Deb, S. Fong, X. He, Y. X. Zhao, "From swarm intelligence to metaheuristics: Nature-inspired optimization algorithms," *Computer*, vol. 49, no. 9, pp. 52–59, Sep. 2016.
[46] X.-S. Yang and X. He, "Firefly algorithm: Recent advances and applications," 2013, *arXiv:1308.3898*. [Online]. Available: http://arxiv.org/abs/1308.3898
[47] X.-S. Yang, S. S. Sadat Hosseini, and A. H. Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect," *Appl. Soft Comput.*, vol. 12, no. 3, pp. 1180–1186, Mar. 2012.
[48] L. Zhang, L. Liu, X.-S. Yang, and Y. Dai, "A novel hybrid firefly algorithm for global optimization," *PLoS ONE*, vol. 11, no. 9, Sep. 2016, Art. no. e0163230.
[49] A. Saeed, S. H. Ab Hamid, and M. B. Mustafa, "The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review," *Appl. Soft Comput.*, vol. 49, pp. 1094–1117, Dec. 2016.
[50] A. Nayyar and R. Singh, "Ant colony optimization—Computational swarm intelligence technique," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, New Delhi, India, Mar. 2016, pp. 1493–1499.
[51] A. D. N. Nayyar and N. G. Le Nguyen, *Advances in Swarm Intelligence for Optimizing Problems in Computer Science*. Boca Raton, FL, USA: CRC Press, 2018.
[52] A. Nayyar, D. N. Le, and N. G. Nguyen, "Advances in swarm intelligence and machine learning for optimization problems in image processing and data analytics," *Recent Patents Comput. Sci.*, vol. 12, no. 4, pp. 248–249, 2019.
[53] A. Nayyar, S. Garg, D. Gupta, and A. Khanna, "Evolutionary computation: Theory and algorithms," in *Advances in Swarm Intelligence for Optimizing Problems in Computer Science*. Boca Raton, FL, USA: CRC Press, 2018, pp. 1–26.
[54] G. K. Durbhaka, B. Selvaraj, and A. Nayyar, "Firefly swarm: Metaheuristic swarm intelligence technique for mathematical optimization," in *Data Management, Analytics and Innovation*. Singapore: Springer, 2019, pp. 457–466.
[55] A. Nayyar and N. G. Nguyen, "Introduction to swarm intelligence," in *Advances in Swarm Intelligence for Optimizing Problems in Computer Science*. London, U.K.: Taylor & Francis, 2018, pp. 53–78.
[56] C. Diwaker, P. Tomar, A. Solanki, A. Nayyar, N. Z. Jhanjhi, A. Abdullah, and M. Supramaniam, "A new model for predicting component-based software reliability using soft computing," *IEEE Access*, vol. 7, pp. 147191–147203, 2019.
[57] M. Gheisari, D. Panwar, P. Tomar, H. Harsh, X. Zhang, A. Solanki, A. Nayyar, and J. A. Alzubi, "An optimization model for software quality prediction with case study analysis using MATLAB," *IEEE Access*, vol. 7, pp. 85123–85138, 2019.
[58] A. Nayyar, *Instant Approach to Software Testing: Principles, Applications, Techniques, and Practices*. New Delhi, India: BPB Publications, 2019.
[59] T. Yifei, Z. Meng, L. Jingwei, L. Dongbo, and W. Yulin, "Research on intelligent welding robot path optimization based on GA and PSO algorithms," *IEEE Access*, vol. 6, pp. 65397–65404, 2018, doi: 10.1109/ACCESS.2018.2878615.
[60] X. Hongxin, Y. Bai, H. Hu, T. Xu, and H. Liang, "A novel hybrid model based on TVIW-PSO-GSA algorithm and support vector machine for classification problems," *IEEE Access*, vol. 7, pp. 27789–27801, 2019.
[61] Q. Al-Tashi, S. J. Abdul Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Binary optimization using hybrid grey wolf optimization for feature selection," *IEEE Access*, vol. 7, pp. 39496–39508, 2019.
[62] I. Hazim, M. S. Kiran, and M. Gunduz, "A novel candidate solution generation strategy for fruit fly optimizer," *IEEE Access*, vol. 7, pp. 130903–130921, 2019.
[63] H. Rico-Garcia, J.-L. Sanchez-Romero, A. Jimeno-Morenilla, H. Migallon-Gomis, H. Mora-Mora, and R. V. Rao, "Comparison of high performance parallel implementations of TLBO and Jaya optimization methods on manycore GPU," *IEEE Access*, vol. 7, pp. 133822–133831, 2019.
[64] M. I. Abdelwanis, A. Abaza, R. A. El-Sehiemy, M. N. Ibrahim, and H. Rezk, "Parameter estimation of electric power transformers using coyote optimization algorithm with experimental verification," *IEEE Access*, vol. 8, pp. 50036–50044, 2020.
[65] L. Tong, X. Li, J. Hu, and L. Ren, "A PSO optimization scale-transformation stochastic-resonance algorithm with stability mutation operator," *IEEE Access*, vol. 6, pp. 1167–1176, 2018.
[66] A. Kumar, P., Srikanth, A., Nayyar, "A novel simulated-annealing based electric bus system design, simulation, and analysis for Dehradun Smart City," *IEEE Access*, vol. 8, pp. 89395–89424, 2020, doi: 10.1109/ACCESS.2020.2990190.
[67] O. Räihä, "A survey on search-based software design," *Comput. Sci. Rev.*, vol. 4, no. 4, pp. 203–249, Nov. 2010.
[68] M. Harman and P. McMinn, "A theoretical and empirical study of search-based testing: Local, global, and hybrid search," *IEEE Trans. Softw. Eng.*, vol. 36, no. 2, pp. 226–247, Mar./Apr. 2010.
[69] A. Arcuri and X. Yao, "A novel co-evolutionary approach to automatic software bug fixing," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.)*, Jun. 2008, pp. 162–168.
[70] C. L. Simons, I. C. Parmee, and R. Gwynllyw, "Interactive, evolutionary search in upstream object-oriented class design," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 798–816, Nov. 2010.
[71] M. Harman, "The current state and future of search based software engineering," in *Proc. Future Softw. Eng. (FOSE)*, vol. 7, May 2007, pp. 342–357.
[72] A. C. Schultz, J. J. Grefenstette, and K. A. De Jong, "Test and evaluation by genetic algorithms," *IEEE Expert*, vol. 8, no. 5, pp. 9–14, Oct. 1993.
[73] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
[74] J. Clarke, J. J. Dolado, M. Harman, R. Hierons, B. Jones, M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper, and M. Shepperd, "Reformulating software engineering as a search problem," *IEE Proc. - Softw.*, vol. 150, no. 3, pp. 161–175, Jun. 2003.
[75] J. H. Holland, *Adaption in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1975.
[76] M. Harman, "Automated test data generation using search based software engineering," in *Proc. 2nd Int. Workshop Automat. Softw. Test (AST)*, vol. 7, May 2007.
[77] B. Uzun and B. Tekinerdogan, "Model-driven architecture based testing: A systematic literature review," *Inf. Softw. Technol.*, vol. 102, pp. 30–48, Oct. 2018.
[78] S. Rajvir, "Review of model based approach for automating the test case generation for Object Oriented Systems," *Int. J. Eng. Comput. Sci.*, vol. 4, no. 6, pp. 12774–12780, 2015.
[79] S. Mahesh and R. Kumar, "UML behavioral model based test case generation: A survey," *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 4, pp. 1–13, 2013.
[80] A. V. K. Shanthi and D. G. M. Kumar, "Automated test cases generation for object oriented software," *Indian J. Comput. Sci. Eng.*, vol. 2, no. 4, pp. 543–546, 2011.
[81] S. Supavita, "Object oriented software and UML-based testing:A survey report," CiteSeerX-Sci. Literature Digit. Library Search Engine, Tech. Rep., 2009.
[82] B. Rumpe, "Model-based testing of object-oriented systems," in *Proc. Int. Symp. Formal Methods Compon. Objects*. Berlin, Germany: Springer, 2002, pp. 380–402, 2002.
[83] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms–A survey," *Swarm Evol. Comput.*, vol. 44, pp. 695–711, Feb. 2019.
[84] R. K. Sahoo, S. K. Nanda, D. P. Mohapatra, and M. R. Patra, "Model driven test case optimization of UML combinational diagrams using hybrid bee colony algorithm," *Int. J. Intell. Syst. Appl.*, vol. 9, no. 6, pp. 43–54, Jun. 2017.
[85] Z. Meng, J.-S. Pan, and K.-K. Tseng, "PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization," *Knowl.-Based Syst.*, vol. 168, pp. 80–99, Mar. 2019, doi: 10.1016/j.knosys.2019.01.006.

[86] C. Grosan and A. Abraham, "Hybrid evolutionary algorithms: Methodologies, architectures, and reviews," in *Studies in Computational Intelligence (SCI)*, vol. 75. Berlin, Germany: Springer, 2007, pp. 1–17.
[87] D. Molina, J. Poyatos, J. Del Ser, S. García, A. Hussain, and F. Herrera, "Comprehensive taxonomies of Nature- and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis and recommendations," 2020, *arXiv:2002.08136*. [Online]. Available: http://arxiv.org/abs/2002.08136

MADHUMITA PANDA received the M.Tech. degree in computer science and engineering from the National Institute of Technology, Rourkela, India. She is currently pursuing the Ph.D. degree in CSE and IT with North Orissa University, Baripada. She has more than ten publications in various reputed International journals and conferences. Her current research interests include pattern recognition, software engineering, model-based optimization, and biometrics.

SUJATA DASH (Member, IEEE) was a Visiting Professor with the Computer Science Department, University of Manitoba, Canada. She has been teaching and guiding students for more than two and a half decades. She is currently an Associate Professor of computer science with the Department of Computer Science, North Orissa University, Baripada, India. She has published more than 160 technical papers in international journals, proceedings of international conferences, and edited book chapters of reputed publishers, such as Springer, Elsevier, the IEEE, and IGI Global, USA. She holds eight national and international patents and published many text books, monographs, and edited books. She has visited many countries and delivered keynotes, invited speech, and chaired many special sessions with the International conferences in India and abroad. Her current research interests include machine learning, data mining, big data analytics, bioinformatics, soft computing, and intelligent agents. She is a member of the international professional associations, such as ACM, IRSS, CSI, IMS, OITS, OMS, IACSIT, IST, and the IEEE. She was a recipient of the Titular Fellowship from the Association of Commonwealth Universities, U.K. She serves as a member of the editorial board for around ten international journals. She also serves as a Reviewer for around 15 international journals, which include World Scientific, Bioinformatics, Springer, IEEE Access, Inderscience, and Science Direct publications.

ANAND NAYYAR (Senior Member, IEEE) received the Ph.D. degree in wireless sensor networks (computer science) from Desh Bhagat University, in 2017. He is currently with the Graduate School, Duy Tan University, Da Nang, Vietnam. He has published more than 300 research papers in various national and international conferences and international journals (Scopus/SCI/SCIE/SSCI Indexed). He was also an ACM Distinguished Speaker. He has authored/coauthored cum edited 25 books of computer science. He was associated with more than 400 international conferences as a programme committee/advisory board/review board member. He holds two Patents in the Internet of Things and speech processing. His current research interests include wireless sensor networks, MANETS, swarm intelligence, cloud computing, the Internet of Things, blockchain, machine learning, deep learning, cyber security, network simulation, and wireless communications. He was a Senior Member and a Life Member of more than 50 Associations. He received several certified professional with more than 75 Professional certificates from CISCO, Microsoft, Oracle, Google, Beingcert, EXIN, GAQM, Cyberoam, and so on. He received more than 20 Awards for the Teaching and Research—Young Scientist, the Best Scientist, the Young Researcher Award, the Outstanding Researcher Award, and the Publons-Top 1% Reviewer Award (Computer Science and Engineering and Cross-Fields). He serves as the Editor-in-Chief for textitInternational Journal of Smart Vehicles and Smart Transportation (IJSVST) (IGI-Global), USA.

MUHAMMAD BILAL (Member, IEEE) received the B.Sc. degree in computer systems engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2008, the M.S. degree in computer engineering from Chosun University, Gwangju, South Korea, in 2012, and the Ph.D. degree in information and communication network engineering from the School of Electronics and Telecommunications Research Institute (ETRI), Korea University of Science and Technology, in 2017. He was a Postdoctoral Research Fellow with the Smart Quantum Communication Center, Korea University, Seoul, South Korea, in 2017. He is currently an Assistant Professor with the Division of Computer and Electronic Systems Engineering, Hankuk University of Foreign Studies, Yongin, South Korea. His research interests include design and analysis of network protocols, network architecture, network security, the IoT named data networking, blockchain, cryptology, and the future internet. He served as a Reviewer for various international journals, including the IEEE.

RAJA MAJID MEHMOOD (Member, IEEE) received the B.S. degree in computer science from Gomal University, Pakistan, in 2004, the M.S. degree in software technology from Linnaeus University, Sweden, in 2010, and the Ph.D. degree in computer engineering from the Division of Computer Science and Engineering, Chonbuk National University, South Korea, in 2017. From April 2011 to February 2014, he was a Lecturer with the Software Engineering Department, King Saud University, Saudi Arabia. He was also a Research Professor with the Department of Brain and Cognitive Engineering, Korea University. He is currently an Assistant Professor with the Information and Communication Department, School of Electrical and Computer Engineering, Xiamen University Malaysia. His main research interests include affective computing, brain-computer interfaces, information visualization, image processing, pattern recognition, and multitask scheduling.