Automated Test Data Generation Using a Combination of Firefly Algorithm and Asexual Reproduction Optimization Algorithm

Amir Hossein Damia
Faculty of Computer Engineering
K. N. Toosi University of Technology
Tehran, Iran
damiaa@email.kntu.ac.ir

Mehdi Esnaashari*
Faculty of Computer Engineering
K. N. Toosi University of Technology
Tehran, Iran
esnaashari@kntu.ac.ir

Received: 2020/08/05 Revised: 2020/09/14 Accepted: 2020/09/28

***Abstract*— Software testing is an expensive and time-consuming process. These costs can be significantly reduced using automated methods. Recently, many researchers have focused on automating this process using search algorithms. Many different methods have been proposed, all of which using a means of heuristic or meta-heuristic search algorithms. The main problem with these methods is that they are usually stuck in local optima. In this paper, to overcome such a problem, we have combined the firefly algorithm (FA) and asexual reproduction optimization algorithm (ARO). FA is a bio-inspired algorithm that is very efficient at exploitation and local searches; however, it suffers from poor exploration and is prone to local optima problem. On the other hand, ARO can be used for escaping from local optima. For this combination, we have inserted ARO into the steps of FA for increasing the population diversity. We have utilized this combination for automatic test case generation with the aim of covering all finite paths of the control flow graph. To evaluate the performance of the proposed method, we have utilized it for generating test cases for a number of programs. Results have indicated that, while giving similar results in terms of the test coverage, the proposed method is significantly better than the existing state of the art algorithms in terms of the number of fitness evaluations. Compared algorithms are FA, ARO, traditional genetic algorithm (TGA), adaptive genetic algorithm (AGA), adaptive particle swarm optimization (APSO), hybrid genetic tabu search algorithm (HGATS), random search (RS), differential evolution (DE), and hybrid cuckoo search and genetic algorithm (CSGA).**

Keywords*— *Software Test; Test Data Generation; Search Algorithms; Firefly Algorithm; Asexual Reproduction Optimization Algorithm.

1. INTRODUCTION

Software testing is one of the most important ways of software quality assurance. When a software is developed, it should be reviewed and be tested. The test stage is the most sensitive stage of the software development process and embraces half of the software development costs [1]. Despite its necessity, testing does not add any new functionality to the software, and thus, great efforts have been made to reduce its costs by automating it. In the last decade, various methods have been introduced for automatic software testing, all of which aimed at maximizing error detection by producing the least number of test data. In the process of generating test data, we need a clue for determining how much test data must be generated [2]. This clue is referred to as the coverage criterion. Different criteria have been defined in literature so far, and selecting one is an engineering problem in which one has to consider his/her time and budget. Coverage criteria are classified into the following four classes: input space partitioning, graph coverage, logic coverage, and syntax-based [2].

According to the selected coverage criterion and its class, input data for the test generation process is determined. For instance, for input space partitioning, this data is the number and types of input arguments of the software under the test (SUT). Or, for graph coverage class, input data is the control flow graph of the SUT. Using this input data, the problem is to find a set of test cases that can satisfy the selected coverage criterion. This is a search problem within a complex and large search space [3]. Therefore, heuristic and meta-heuristic search algorithms can be utilized here [4]. For this paper, we have considered the "Path Coverage" criterion, which is a criterion reside in the graph coverage class. The specific problem is to find a test set that can cover all finite paths within the control flow graph of the SUT. To solve this problem, a combination of the firefly algorithm and the asexual reproduction optimization algorithm has been used. FA is a swarm intelligence algorithm that is inspired by the behavior of fireflies. It is one of the most successful yet low-cost algorithms. However, it may trap in local [5, 6]. To overcome this problem, in this paper, we have suggested combining FA witch ARO [7]. In this combination, ARO has been used to diversify the population of FA. The ́irefly algorithm is selected due to its power in solving the modern numerical optimization problems, in particular for the NP-hard problems [1] [2]. As compared to particle swarm optimization and genetic algorithm techniques [1], the Firefly Algorithm reduces the overall computational effort by 86% and 74%, respectively [2]. On the other hand, the asexual reproduction optimization algorithm is considered for its high speed and the absence of any additional parameters [3].

The performance and efficiency of the proposed method have been evaluated using a number of SUTs. Results have been compared with the traditional FA, ARO, AGA [8], TGA [9], APSO [10], HGATS [11], RS (it is the simplest algorithm which randomly generates a large number of possible solutions

and chooses better solutions), DE [12], and CSGA [13] in terms of a number of evaluations. A significant advantage of the proposed method over existing methods in terms of the aforementioned criteria have been demonstrated in these results.

The rest of the paper is organized as follows. Section 2 discusses related work. FA and ARO algorithms are briefly presented in Section 3. The proposed method is elaborated in Section 4. Section 5 provides experimental results. Finally, Section 6 concludes the paper.

2. Related Works

There are different methods for designing test data that can be divided into two methods: white box (structural test) and black box (functional test) [14]. In the white box approach, the program code is used to design the test and we are dealing with the internal mechanism of a system, while in the black box approach it focuses only on the outputs of the SUT. This method, which is based on requirements specifications, does not require code execution and can help identify any ambiguities and inconsistencies in the requirements specifications. Among the black box solutions, we can mention the model-based test which has been used for many years in industrial and academic research [15]. This solution, by analyzing the model of the system under test and considering the coverage criterion, is able to systematically produce test items that cover certain features of the model.

From the point of view of algorithm type, test data production algorithms can be divided into 3 categories [16]:

- Random
- Search based (white box testing or structural testing(
- Model based (black box test)

2-1. Random

Random testing was one of the first works in this field and due to the development of other methods, this method has become very inefficient in recent years. The purpose of this paper is to compare this method with other methods. Random testing is the easiest approach for test case generation. Random testing generates test data for any type of program, as it is independently generated from their operational profile. It is not able to cover all types of faults since the random generation of test case may not execute some statements having faults. The probability of finding at least one error in software by random testing depends only on the number of test points [17]. Random testing chooses test data set randomly from uniform distribution and perform testing using these test data set. Random testing program is viewed as a worst case of program testing because of its inability to find failure in the system. A mixed final testing was recommended that merges random testing with value based testing [18]. The easiest and cheaper technique for test data generation is the random testing which require less effort and time for test data generation [19].

2-2. Search based

Multi-population genetic algorithm has been introduced in recent years as an improved genetic algorithm, which has shown good performance. The multi-population approach can prevent the algorithm from being trapped in local minimums and premature convergence, on the other hand, the multi-population model can improve the quality of the solutions and improve the speed of genetic algorithm evolution. In this method, the multi-population genetic algorithm is used to obtain the optimal solution, since two child populations and one main population are used and the child populations run in parallel. Experiments have shown that this method improves the convergence rate, search time, and percentage of coverage and is better than single population genetic algorithm and random search [20]. There are two different ways of adjusting parameter values (recombination rate and mutation rate) of the genetic algorithm: adjusting parameter values before the optimization process, or by dynamically adjusting parameters during execution. In this method, the genetic algorithm improved by maintaining population diversity is used to generate test data that dynamically obtains the recombination rate and mutation rate of the chromosomes with the similarity between the chromosomes and the amount of chromosomes in each step of the algorithm. Genetic algorithm operators are essential to obtain the next generation of a population and to replicate evolution. The classical genetic algorithm encounters the recession phenomenon in its later stages, with constant values of recombination rate and mutation due to population diversity. In the recombination operator, if the rate of this operator is high, the chromosome suitability values may be easily corrupted, and if the rate of this operator is low, no new offspring may be created, so this rate is better given the degree of population variation and the average fitness of the population chromosomes in each calculate the stage of implementation. Experimental results show that this method is more effective than similar methods and random method for path testing. Although the programs selected in this article are in C, this method can be used in another language [8]. Mack Mann and Pradeep Tamar introduced a genetic algorithm-based method for generating software test data and their results were compared with the stochastic method. In this paper, the impact of early population on the efficiency of genetic algorithm is investigated. Their experiments showed that their proposed method is more efficient than the random method and requires less time to generate software test data, and by increasing the initial population size, more search space can be created by increasing diversity, making it less likely the algorithm should be local optimized [21]. Rijwan Khan introduced a method for the automated generation of software test data by combining genetic algorithm and cuckoo search algorithm. Their goal was to reduce the time and cost of producing test data. Cuckoo search algorithms have been used to improve chromosomes. Their experiments have shown that the combination of the two algorithms is better than applying each of them separately [13].In this paper [22], the genetic algorithm and the simulated annealing algorithm are used to automate the production of test data based on the path coverage criteria and their results are compared. Their results show that the genetic algorithm is simulated more efficiently than the simulated annealing algorithm by correctly adjusting the parameters and achieving maximum coverage in the least number of iterations.

In [23], a method for generating test data based on the combination of genetic algorithm and particle swarm optimization algorithm for automating test data generation automation based on path coverage criteria is presented. The efficiency of the proposed methods has been analyzed with programs of different and complex sizes. Finally, this method compares the combination with the genetic algorithm and the particle swarm optimization algorithm, and they have shown

that their combination works better than any of these methods. Another method for generating dynamic test data is based on particle swarm optimization algorithm, which defines the condition of the fitness function for each statement, and only the conditional statements are considered in this method. This method works better than random search and tabu search [24, 25, 26]. It is presented in a way to generate test data using cuckoo search algorithms and tabu search. Tabu search is used to reduce the execution time of the algorithm. This method is better in terms of execution time and performance than particle swarm optimization algorithms and bees algorithm [27, 28].

Sahin and Akay have presented a comparison between important meta-heuristics algorithms used for automatic test data generation [29]. These algorithms have been compared based on different fitness functions (path-based, dissimilarity-based and approximation level + branch distance). This is due to the fact that different fitness functions affect the behavior of the algorithm in the search space. Results have shown that meta-heuristics strategies were very well suited for generating test data.

Kumar et. al. have introduced a method for automatically generating test data using a combination of GA and PSO [30]. Their goal was to overcome the weaknesses of both algorithms. They designed a new fitness function based on the concept of dominance relations, branch weight and branch distance to a better search. Their proposed method have been compared with GA, PSO, ant colony optimization, and differential evolution based on two criteria; average number of generations and average percentage of coverage. The results of their experiments have shown that hybrid PSO-GA gives better results compared to the compared algorithms in the field of test data generation.

Jain et. al. have introduced a new 2-step inharmonious approach based on GA and PSO to class testing using data flow criteria [31]. A set of classes are further tested to study performance of the proposed method in terms of the percentage of coverage and the execution time. The results of their experiments have shown that their proposed method is better than random method in terms of coverage ratio achieved and iterations performed.

In [32] using a combination of firefly algorithm and graph reduction, the standard firefly algorithm has been extended to generate optimal discrete and independent paths for software testing. The proposed approach tries to minimize the number of test data by optimizing the test paths for test data. Their results showed that firefly algorithm based approach has produced the optimal paths below a given number of independent paths and, it can minimize the test efforts and provides the best critical test paths.

2-3. Model based

In this method, software systems models are used to extract the test set. In this type of method, the focus is on behavioral testing, a function, or black box that tests the program based on observable input and output behavior. In this method, SUT is considered as a black box testing that receives inputs and generates outputs. SUT has an internal mode that changes with input processing and output generation. The model describes possible input and output sequences at a certain level of abstraction and links to implementation through a connection. A selection algorithm extracts test data from the model. The use of a test criterion based on a test hypothesis justifies the accuracy of the selection.

This method has been used in [33]. The authors have utilized UML state machine diagrams for automatically generating test scenarios for concurrent and composite states. The firefly algorithm has been used only for prioritizing test scenarios.

3. BRIEF EXPLANATIONS OF USING ALGORITHMS IN THE PAPER

Inspired by a natural phenomenon, meta-heuristic algorithms allow a very large search space to be intelligently explored. What is clever about this is that meta-heuristic algorithms do not navigate the entire search space; instead, they only navigate the part of the space where there is a good chance that a good enough point exits there. Brief descriptions of FA and ARO algorithm are given in the following sections.

3-1. Asexual Reproduction Optimization Algorithm

The asexual reproductive optimization is a meta-heuristic search method, in which, a single member (parent chromosome) is randomly generated and is then evaluated. The following operations are repeated:

First, a string of genomes in the parent chromosome is randomly selected and mutated. The length of the selected string (g) is randomly selected within the range [1,L], where L is the length of the parent chromosome. The result of this mutation is referred to as the larva. Next, with a probability of pc, defined in (1) given below, the parent and the larva chromosomes are undergo a uniform recombination operation. The recombinant result is referred to as the bud chromosome.

$$pc = \frac{1}{1+\ln g} \quad (1)$$

ARO is greedy in the sense that the child produced, either the larva or the bud, replaces his parent only if he is more fit. Main features of this algorithm are its high speed and the absence of any additional parameters.

3-2. Firefly Algorithm

The firefly algorithm is designed by modeling the luminosity characteristics of firefly worms [34]. To simplify the definition of this algorithm, the following three assumptions have been considered:

1. All fireflies are of a kind and attract pairs to each other regardless of gender.
2. Attractiveness is relative to their brightness. So for each pair of firefly worms, a worm that has less light is attracted to a worm that has more light. The absorption power is proportional to the intensity of their light and the intensity of the light decreases with increasing distance between the two worms. If the light intensity of the two worms is similar, their movement would be random.
3. The brightness of firefly worms is determined by the value of the objective function.

Since the attractiveness of a firefly worm is commensurate with the intensity of light being seen by nearby firefly worms, we can define the attractiveness of the worm i from the point of view of the worm j according to (2), given below.

$$\beta_{i,j} = \beta_0 \, e^{-\gamma r_{i,j}^2} \qquad (2)$$

Here, β_0, is the attractiveness of the firefly worm i at distance $r = 0$ and γ is the light absorption coefficient. γr^2 can be replaced by other functions such as γr^m if $m > 0$.

The movement and absorption of worm i into worm j is determined according to (3), given below.

$$x_i = x_i + \beta_{i,j}\left(x_j - x_i\right) + \alpha \qquad (3)$$

x_i and x_j represent worms i and j respectively. α is a random value, within the range [0, 1].

4. The Proposed Method

Premature convergence is an important issue in the firefly algorithm. Studies have shown that there is a direct relationship between early and premature convergence and the population diversity [35, 36, 6, 37]. In the proposed method, we have modified FA so that its population is diversified using ARO. In what follows, we will first formulate the problem, and then, explain the proposed algorithm is detail.

4-1. Formulation of the problem

In the process of producing structural test data, we need criteria to determine how complete the generated data is for testing the software under the test. These criteria, referred to as coverage criteria, are defined based on the structure of the SUT. For example, the branch coverage criterion is one of the criteria in which the goal is to cover all branches of the SUT. Another criterion is the expression coverage, which aims to execute all program expressions. In this paper, we focus on the path coverage criterion, in which the aim is to cover all finite paths in the SUT.

Finding all finite paths of the SUT can be carried out using the control flow graph (CFG) of the SUT. Every SUT can be represented by its CFG $= (N, E, s, e)$, which is a directional graph. N is equal to the set of nodes, and E is equal to the set of directional edges a_{ij}, $s \in N$ is the initial node and $e \in N$ is the terminating node. Each node represents a linear sequence of program calculations being tested. Each edge a_{ij} represents the transfer of SUT execution control from node n_i to node n_j. For example, Fig.1 shows the CFG associated with the SUT.

The specific problem in this paper is to automatically generate input data that can test and cover all finite paths in the SUT, with the goal of minimizing the number of test cases. This is proved to be an NP-hard problem [43].

4-2. Population Initialization

The first step in the FA algorithm is creating the first population randomly. The population consists of a number of firefly worms. Suppose the number of input parameters is m and the number of paths in the CFG is n. Then a worm which can represent a test set for a given SUT can be defined as: $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]$, $x_{ik} = \left[a_{ik,1}, a_{ik,2}, \ldots, a_{ik,m}\right]$. Each x_{ik} is a sub-section of the worm which represents a test case, with the hope of covering path k in the CFG. Since there are n paths, thus k∈{1,2,…,n}. Each a_(ik,l) represents an input parameter of the SUT, thus $k \in \{1, 2, \ldots, n\}$. Fig.2 shows an example of a worm. In other words, each worm represents a complete test set, which may or may not cover all paths within the CFG. According to this structure, a number of random worms are generated to form the initial population.

4-3. Fitness Function

After generating the initial random population, to determine the fitness of each worm, it is necessary to run the SUT n times, each time with a specific x_{ik}. Executing SUT with a specific x_{ik} indicates which path within the SUT is covered with the given input parameters $\left[a_{ik,1}, a_{ik,2}, \ldots, a_{ik,m}\right]$. The result of these executions determines to what extent x_i covers paths in the CFG. The fitness of each worm is then calculated according to (4), given below.

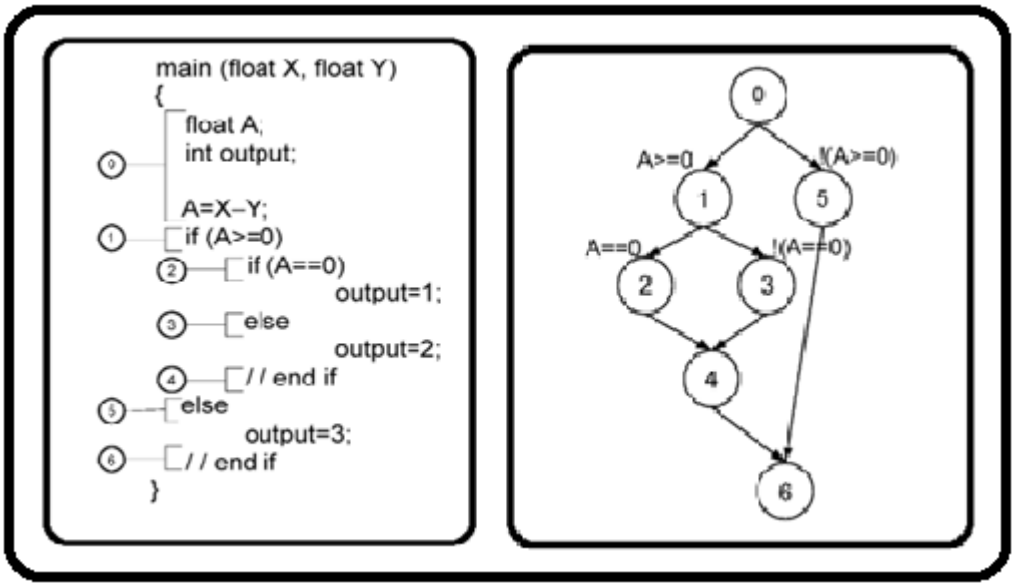

Fig 1: A sample SUT and its corresponding CFG.

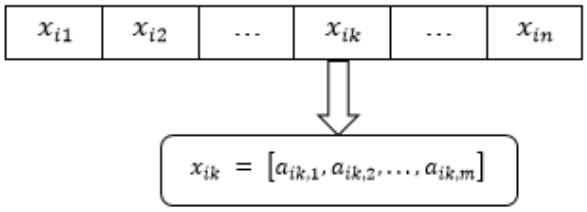

Fig 2: A firefly worm which can be used for the test data generation problem

$$\text{Absolute Error} = |\text{Number of Covered Paths} - n|$$

$$\text{Error Percentage} = \frac{\text{Absolute Error}}{n} \times 100$$

$$\text{Fitness}(x_i) = f_i = 100 - \text{Error Percentage} \quad (4)$$

4-4. Modifications Applied to the Basic FA

The main modification is to ensure diversity in the population. Diversity can be defined according to Euclidean distance between individuals. However, if we only consider the distance for diversifying the population, we may neglect the fitness information of individuals. Therefore, in this paper we consider a population diversity metric that considers both distance and fitness information [38].

Definition 1: Dissimilarity: The Euclidean distance d_{ij} between the two worms x_i and x_j is considered as their dissimilarity, and is calculated according to (5), given below.

$$d_{ij} = |x_i - x_j| = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2} \ , (i, j = 1,2, \ldots, n) \quad (5)$$

The distance of the worm x_i with the whole population can be calculated according to (6).

$$d_i = \sum_{j=1}^{ps} d_{ij} \quad (6),$$

where *ps* is the population size. Then the distance of the population can be calculated according to (7), as given below.

$$D = \sum_{i=1}^{ps} d_i \quad (7)$$

Definition 2: The average fitness of the population can be calculated according to (8).

$$f = \frac{1}{ps} \sum_{i=1}^{ps} f_i \quad (8)$$

According to the values of *D* and *f*, a diversification process is executed in the following situations:

- *D* is less than a predetermined fixed threshold
- The value of *D* is not changed in subsequent number of generations
- The value of *f* is not changed in subsequent number of generations

If a diversification is required, then ARO algorithm is utilized for this purpose. Each individual x_i in the population is passed into the ARO algorithm. It is first mutated to make a larva, and then, with a probability of *pc* is recombined with the larva to become the bud. Either of the larva or the bud is then replaced with the x_i in the population.

4-5. The Complete Algorithm

Algorithm 1 presents the pseudo code of the proposed FA-ARO algorithm. The initial population evolves according to the FA-ARO strategy, until one of the following two criteria is met:

1. One firefly worm is discovered which covers all *n* paths of the given CFG.

2. Maximum number of evaluations is reached.

5. Experimental Results

5-1. Evaluation Metrics

In this paper, we have utilized following evaluation criteria for evaluating the performance of the proposed method in comparison to existing state of the art methods:

- Coverage Ratio: Defined as the fitness of the best individual.

- FEvals: Number of fitness evaluations before finding the answer.

5-2. Benchmark Programs

To conduct experiments, several benchmark programs (SUTs) have been selected. The purpose of these benchmarks is to cover a variety of structures such as loops and conditions. For each SUT, we have described why it is considered here.

SUT1- Quadratic Equation: In algebra, a quadratic function, a quadratic polynomial, a polynomial of degree 2, or simply a quadratic, is a polynomial function with one or more variables in which the highest-degree term is of the second degree [39]. This program takes three inputs. The number of finite paths within its CFG is 4.

This SUT is utilized for checking if the proposed method is able to consider conditionals with specific values. The SUT has different paths for positive, negative, and zero deltas. The toughest path is the one with delta = 0, since there are very few cases in which delta can be equal to zero.

SUT2- Triangle Classification: Triangles can be classified according to the lengths of their sides [40]:

- All sides the same length (equilateral).
- Two sides of equal length (isosceles).
- All sides of different lengths (scalene).

This program takes three inputs. The number of finite paths within its CFG is 3. This SUT is a simple one for which test cases could be found easily. However, it is a well-known benchmark in the field and thus, we have considered it here.

SUT3- Binary Search: It is a technique for finding a numeric value from a set of ordered numbers. This method reduces the search range by half at each step, so the target is either soon found or it becomes clear that the search value is not in the list [41]. This program takes two inputs: a sorted array and the target value. Number of finite paths within the CFG of this program depends on the length of the input array. This SUT is considered here with the purpose of testing if the proposed method is able to provide test cases for recursive calls.

Algorithm 1 The proposed FA-ARO algorithm.

```
1: Inputs
   The SUT, n, m, ps (population size), max_ evaluations, diversity_threshold, β0, α, γ
   number of iteration (ARO algorithm), rounds of the run algorithm
2: Output
   Set of test data for the SUT
3: Population = [ ]
4: Sbest = [ ]
5:
6 number of evaluations = 0
7: g = 0
8: For i = 1 to ps do
9:       x_i = MakeRandomWorm(n, m)
10:      compute f_i according to the equation (4)
11:Compute f according to the equation (8)
12:
13: While  (not all paths are covered) or (number of evaluations <  max_ evaluations )
14:      For i=1 to ps do
15:          For j = 1 to ps do
16:              If f_j > f_i
17:                   population= UpdatePosition (Population,i,j,β0 ,γ , α )
18:          End
19:     End
20:     fitness = EvaluationPopulation (Population)
21:     g=g +1
22:     If g ==     rounds of the run algorithm:
23:
24:          g=0
25:          Compute the f values according to the equation (8)
26:
27:          Compute the D values according to the equation (7)
28:
29:          If ( D <  diversity_threshold) or
                 (the value of D is not changed in subsequent generations) or
                       (the value of f is not changed in subsequent generations)
30:
31:                For i to ps do
32:                     x_i= ARO(x_i, number of iteration)
33:          End
34:      End
35:      number of evaluations = number of evaluations + population size
35: End
```

SUT4- Fibonacci: Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1 [42]. This program takes one input. The number of finite paths within its CFG 4. The aim of considering this SUT here is to check for loops.

SUT5- Find Area of a Triangle: This program takes three inputs. The number of finite paths within its CFG is 7. In this program, there are paths that must produce certain values that must pass nested conditions.

SUT6- A Simple Calculator: This Program is a simple calculator that can add, subtract, multiply and divide two given numbers. This program takes three inputs. The number of finite paths within its CFG is 10. In the calculator program, nesting functions is considered.

5-3. Parameter Settings

Table 1 lists the various operator and parameter settings used for experiments in this paper.

5-4. Experiments

This experiment aims at showing the superiority of the proposed method compared with other existing methods in terms of both coverage ratio and the number of fitness evaluations. The experiment has been carried out over all six benchmark programs; SUT1 ~ SUT6. Each algorithm has been executed 50 times to obtain statistically significant results. Results of the experiment have been reported in Table 2 and in terms of FEvals and coverage ratio respectively. Table 2 provides mean, standard deviation, and the P-value (t-test with

TABLE 1: PARAMETER SETTINGS.

Algorithm	Parameter/ Operator	Value	Population Size	Population Representation	Data range
ARO [7]	Crossover type	uniform	1	integer	[-100, 100]
FA [34]	alpha	0.50	40		
	gamma	1			
	$\beta 0$	1			
FA-ARO	alpha	0.50			
	gamma	1			
	$\beta 0$	1			
	Crossover type	uniform			
AGA [8]	Pc0	.9			
	Pm0	.2			
	selection	linear rank			
	Crossover type	One point			
	Mutation type	One point			
	σ	Range of [.4, .6]			
TGA [9]	pc	.85			
	pm	.15			
	selection				
APSO [10]	Wmax	1.0			
	Wmin	.4			
	C1	2			
	C2	2			
HGATS [11]	Pc	.85			
	Pm	.12			
	selection	roulette wheel			
DE [12]	Mutation Scaling Factor: F	1			
	Crossover Constant: CR	.9			
CSGA [13]	pc	.10			
	pm	.85			
	selection	tournament selection			

α=.05) for each algorithm per benchmark programs. In particular, p-value indicates the probability of error in accepting the validity of the observed results. For example, a p-value of 0.0008 for the results of FA algorithm for SUT1, indicates that there is a .08% probability that the observed mean and STD are random. In general, the lower the p-value, the more accurate our result would be.

Coverage ratio for all algorithms and all SUTs were 100%. It could be seen from Table 2 that the proposed FA-ARO algorithm substantially outperforms other existing algorithms in terms of the number of fitness evaluations. The main reasons for superiority of the FA-ARO over existing algorithms are its ability in escaping from local optima, with the help of ARO and its performance in improving individuals of the population with the help of FA. In TGA and AGA, no individual tries to improve itself. The only means of improvement is carried out through the mutation and crossover operations. However, in the proposed algorithm, each worm independently tries to improve. In ARO, in addition to the above mentioned problem, due to the lack of cooperation between individuals, the convergence

TABLE 2: EXPERIMENT RESULTS: FEVAL.

Program / Algorithm		SUT1	SUT2	SUT3	SUT4	SUT5	SUT6
FA	Mean	20847.2	64721.6	3374.6	5450.4	17412.8	12920.4
	Std	1691.42	5596.00	253.77	495.09	1226.69	796.12
	P-value	0.152	0.0008	0.0	0.0	2e-07	2e-05
ARO	Mean	23766.4	72076.5	2949.9	5007.1	13183.9	13988.2
	Std	1952.73	7762.81	340.67	433.96	704.02	696.60
	P-value	0.034	0.001	0.001	1e-06	1e-03	0.0
FAARO	Mean	16120.8	33074.1	1221.1	1621.9	6988.0	6895.6
	Std	1547.12	3177.51	114.98	104.07	439.30	506.55
	P-value	1.0	1.0	1.0	1.0	1.0	1.0
AGA	Mean	16879.2	36292.0	1486.6	2899.8	10464.7	10675.7
	Std	2207.02	4156.05	182.54	208.01	574.80	443.84
	P-value	0.844	0.666	0.391	0.0002	0.001	0.0001
TGA	Mean	23095.2	63338.4	2755.2	6030.5	21945.6	11179.5
	Std	2490.43	7583.93	302.09	494.95	1078.06	547.04
	P-value	0.099	0.011	0.001	2e-08	0.0	0.0001
APSO	Mean	16920.8	38819.2	2517.9	2036.8	8957.5	7474.4
	Std	1558.55	3494.32	207.61	145.23	638.01	360.82
	P-value	0.799	0.394	0.0002	0.106	0.078	0.516
HGATS	Mean	17788.8	47557.6	2073.9	3748.4	18275.0	11901.1
	Std	1589.25	4300.53	203.21	307.84	1617.53	700.86
	P-value	0.599	0.060	0.012	1.34e-05	8e-06	0.0001
RS	Mean	42191.2	102775.6	5610.5	6457.6	65388.8	22805.1
	Std	4148.01	10743.18	549.40	652.66	4257.31	1106.56
	P-value	8e-05	3e-05	0.0	1e-06	0.0	0.0
DE	Mean	18576.0	45760.8	1821.96	3590.4	13689.6	10091.18
	Std	1571.16	4350.41	188.34	285.12	1540.11	420.56
	P-value	0.307	0.060	0.033	0.001	0.001	0.023
CSGA	Mean	19852.0	46161.6	2452.8	4033.6	13879.2	9157.6
	Std	2130.32	4960.34	430.40	270.20	1251.12	485.37
	P-value	0.129	0.040	0.0003	4.062e-06	0.0003	0.162

time to an acceptable answer may be too long. APSO and FA suffer from local optimum traps, and hence, usually are not able to find global optima. DE has many disadvantages, including unstable convergence in the last period and easy to be trapped into local optimum. In CSGA, there is not much improvement over genetic algorithms, and there are still common problems with genetic algorithms in CSGA.

6. CONCLUSION

This paper provides a way for automatically generating test data using a meta-heuristic method that combines the firefly algorithm and asexual reproduction optimization algorithm. The firefly algorithm has a high speed, but it suffers from the problem of premature convergence. On the other hand, the

ARO algorithm is well suited for escaping from local optima. In the combined method, referred to as FA-ARO, ARO was considered as a step within the steps of FA algorithm for diversifying the population. In order to evaluate the efficiency of the proposed method, its results were compared with that of the existing state of the art algorithms in terms of coverage ratio as well as the number of fitness evaluations. It was indicated that the proposed FA-ARO method is significantly better than the existing methods in terms of the number of fitness evaluations while at the same time provides equal test coverage.

REFERENCES

[1] G. J. Myers, C. Sandler, and T. Badgett, The art of software testing, John Wiley & Sons, 2011.

[2] P. Ammann, and J. Offutt, Introduction to software testing. Cambridge University Press, 2016.

[3] P. McMinn, "Search- based software test data generation: a survey". Software testing, Verification and reliability, vol. 14, no. 2, pp. 105-156, 2004.

[4] R. R. Sahoo, and M. Ray, "Metaheuristic techniques for test case generation: a review". Journal of Information Technology Research (JITR), vol. 11, no. 1, pp.158-171, 2018.

[5] N. J. Cheung, X. M. Ding, and H. B. Shen, "Adaptive firefly algorithm: parameter analysis and its application". PLoS One, vol. 9, no. 11, e112634, 2014.

[6] I. Fister, X. S. Yang, and J. Brest, "Memetic self-adaptive firefly algorithm". In Swarm intelligence and bio-inspired computation, Elsevier. 2013, pp. 73-102.

[7] A. Farasat, M. B. Menhaj, T. Mansouri, and M. R. S. Moghadam, "ARO: A new model-free optimization algorithm inspired from asexual reproduction". Applied Soft Computing, vol. 10, no. 4, pp. 1284-1292, 2010.

[8] X. Bao, Z. Xiong, N. Zhang, J. Qian, B. Wu, and W. Zhang, "Path-oriented test cases generation based adaptive genetic algorithm". PloS one, vol. 12, no. 11, 2017.

[9] M. R. Girgis, "Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm". J. UCS, vol. 11, no. 6, pp. 898-915, 2005.

[10] S. Jiang, J. Shi, Y. Zhang, and H. Han, "Automatic test data generation based on reduced adaptive particle swarm optimization algorithm". Neurocomputing, vol. 158, pp.109-116, 2015.

[11] X. Fan, "Test data generation with a hybrid genetic tabu search algorithm for decision coverage criteria". In The 5th International Conference on Computer Engineering and Networks, SISSA Medialab, October 2015, vol. 259, p.008.

[12] S. Varshney, and M. Mehrotra, A differential evolution based approach to generate test data for data-flow coverage. In 2016 International Conference on Computing, Communication and Automation (ICCCA), IEEE, April 2016, pp. 796-801.

[13] [13] R. Khan, M. Amjad, and A. K. Srivastava, "Optimization of automatic test case generation with cuckoo search and genetic algorithm approaches", Advances in Computer and Computational Sciences, Springer, Singapore, 2018, pp. 413-423.

[14] P. Ammann, and J. Offutt, Introduction to Software Testing, 1 ed., United States of America, New York, Cambridge University Press, April 2008.

[15] M. Utting, B. Legeard, F. Bouquet, E. Fourneret, F. Peureux, and A. Vernotte, "Recent advances in model-based testing", In Advances in Computers, vol. 101, Elsevier, January 2016, pp. 53-120.

[16] F. Lonetti, and E. Marchetti, "Emerging software testing technologies". In Advances in Computers, vol. 108, Elsevier, 2018, pp. 91-143.

[17] R. Hamlet, "Random testing", in Encyclopedia of software Engineering,

[18] S. C. Ntafos, and J. W. Duran, "An Evolution of random testing", IEEE Transaction on Software Testing, vol. 10, no. 4, pp. 438-444, July 1984.

[19] D. C. Ince, "The automatic generation of test data", The Computer Journal, vol. 30, no. 1, pp. 63-69, 1987.

[20] N. Zhang, B. Wu, and X. Bao. "Automatic generation of test cases based on multi-population genetic algorithm". Int. J. Multimedia Ubiquitous Eng., vol. 10, no. 6, pp. 113-122, 2015.

[21] M. Mann, P. Tomar, and O. P. Sangwan, "Test Data Generation Using Optimization Algorithm: An Empirical Evaluation". Soft Computing: Theories and Applications. Springer, Singapore, 2018, pp. 679-686.

[22] M. Mann, O. P. Sangwan, P. Tomar, and S. Singh, "Automatic goal-oriented test data generation using a genetic algorithm and simulated annealing". 2016 6th International Conference-Cloud System and Big Data Engineering (Confluence). IEEE, 2016 pp. 83-87.

[23] S. Singla, D. Kumar, H. M. Rai, and P. Singla, "A hybrid PSO approach to automate test data generation for data flow coverage with dominance concepts". International Journal of Advanced Science and Technology, vol. 37, pp. 15-26, 2011.

[24] A. A. Sofokleous, and A.S. Andreou. "Automatic, evolutionary test data generation for dynamic software testing". Journal of Systems and Software, vol. 81, no. 11, pp. 1883-1898, 2008.

[25] G. I. Latiu, O. A. Cret, and L. Vacariu, "Automatic test data generation for software path testing using evolutionary algorithms". 2012 Third International Conference on Emerging Intelligent Data and Web Technologies. IEEE, 2012, pp. 1-8.

[26] A. Windisch, S. Wappler, and J. Wegener, "Applying particle swarm optimization to software testing". Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, 2007, pp. 1121-1128.

[27] P. R. Srivastava, R. Khandelwal, S. Khandelwal, S. Kumar, and S. S. Ranganatha, "Automated test data generation using cuckoo search and tabu search (CSTS) algorithm". Journal of Intelligent Systems, vol. 21, no. 2, pp. 195-224, 2012.

[28] K. Perumal, J. M. Ungati, G. Kumar, N. Jain, R. Gaurav, and P. R. Srivastava, "Test data generation: a hybrid approach using cuckoo and tabu search". International Conference on Swarm, Evolutionary, and Memetic Computing, Springer, Berlin, Heidelberg, 2011, pp. 46-54.

[29] O. Sahin, and , B. Akay "Comparisons of metaheuristic algorithms and fitness functions on software test data generation". Applied Soft Computing, vol. 49, pp.1202-1214, 2016.

[30] S. Kumar, D. K. Yadav, and D. A. Khan, "A novel approach to automate test data generation for data flow testing based on hybrid adaptive PSO-GA algorithm". International Journal of Advanced Intelligence Paradigms, vol. 9, no. 2-3, pp.278-312, 2017.

[31] N. Jain, R. Porwal, S. Kumar, S.Varshney, and M. Saraswat, "Automatic data flow class testing based on 2-step heterogeneous process using evolutionary algorithms". Journal of Statistics and Management Systems, vol. 22, no. 7, pp.1315-1348, 2019.

[32] P. R. Srivatsava, B. Mallikarjun, and X. S. Yang, "Optimal test sequence generation using firefly algorithm". Swarm and Evolutionary Computation, vol. 8, pp.44-53, 2013.

[33] D. P. Mohapatra, "Firefly optimization technique based test scenario generation and prioritization". Journal of Applied Research and Technology, vol. 16, no. 6, pp.466-483, 2018.

[34] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization". International Journal of Bio-inspired Computation, vol. 2, no. 2, pp.78-84, 2010.

[35] B. McGinley, J. Maher, C. O'Riordan, and F. Morgan, "Maintaining healthy population diversity using adaptive crossover, mutation, and selection". IEEE Transactions on Evolutionary Computation, vol. 15, no. 5, pp.692-714, 2011.

[36] Y. Leung, Y. Gao, and Z. B. Xu, "Degree of population diversity-a perspective on premature convergence in genetic algorithms and its markov chain analysis". IEEE Transactions on Neural Networks, vol. 8, no. 5, pp.1165-1176, 1997.

[37] A. E. Eiben, and J. E. Smith, Multimodal problems and spatial distribution. In Introduction to Evolutionary Computing, Springer, Berlin, Heidelberg, 2003, pp. 153-172.

[38] N. Zhang, B. Wu, and X. Bao. "Automatic generation of test cases based on multi-population genetic algorithm". Int. J. Multimedia Ubiquitous Eng., vol. 10, no. 6, pp. 113-122, 2015.

[39] N. Zhang, B. Wu, and X. Bao, "Automatic generation of test cases based on multi-population genetic algorithm". Int. J. Multimedia Ubiquitous Eng, vol. 10, no. 6, pp.113-122, 2015.

[40] A. S., Ghiduk, "Automatic generation of basis test paths using variable length genetic algorithm". Information Processing Letters, vol. 114, no. 6, pp.304-316, 2014.

[41] A. M. Bidgoli, and H. Haghighi, "Augmenting ant colony optimization with adaptive random testing to cover prime paths". Journal of Systems and Software, vol. 161, p.110495, 2020.

[42] G. I., Latiu, O. A. Cret, and L. Vacariu, "September. Automatic test data generation for software path testing using evolutionary algorithms". In 2012 Third International Conference on Emerging Intelligent Data and Web Technologies, IEEE, 2012, pp. 1-8.

[43] M. Harman, and B. F. Jones, "Search-based Software Engineering". Information and Software Technology, vol. 43, pp. 833-839, 2001.

Mehdi Esnaashari received the B.S., M.S., and Ph.D. degrees in Computer Engineering, all from Amirkabir University of Technology, Tehran, Iran, in 2002, 2005, and 2011 respectively. He worked at Iran Telecommunications Research Center as an Assistant Professor from 2012 to 2016. Currently, he is an Assistant Professor in the Faculty of Computer Engineering, K. N. Toosi University of Technology. His research interests include computer networks, learning systems, especially reinforcement learning, and information retrieval.

Amir Hossein Damia received his M.S. degree in the field of software engineering from K. N. Toosi University of Technology in 2020. His research interests include evolutionary algorithms, reinforcement learning, and software testing.