ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2022.0381

A NOVEL APPROACH FOR TEST DATA GENERATION

Gagan Kumar[1] and Vinay Chopra[2]
[1]Department of Computer Science and Engineering, IK Gujral Punjab Technical University, India
[2]Master of Computer Applications, D.A.V. Institute of Engineering and Technology, India

Abstract

Software testing is an essential phase in software design process, accounting for more than half of the total cost due to its rigorous and time-consuming nature. Path test data generation is the most important stage in software testing, and researchers have devised several methods to automate it. In this research, a novel approach based on ant colony optimization and negative selection algorithm (NSA) is projected to automatically create test data for path testing. The most widely used benchmark programs such as triangle classification, dayfinder, minmax and isprime, has been used to test the proposed approach. When compared to random testing, the experimental findings reveal that the proposed method is more efficient in terms of coverage, execution time and more effective in terms of test data creation.

Keywords:

Test Data Generation (TDG), Meta-Heuristic, Artificial Immune Algorithm, ACO, NSA, Path Coverage, Fitness Function

1. INTRODUCTION

Software testing is a pivotal task in software development life cycle. It is an expensive and laborious activity which often accounted as a time-consuming process in any software development life cycle model [1] [2]. It is being used to unfold the bugs and errors out from the software code [3]. Testing can be applied on structural and functional part of the code [4]. Both structural and functional aspects have their own significance, Structural testing is considered as the strongest one out of the two[5].

Structured testing focuses on internal structure of the program based on the fitness criteria opted for the testing. Structure of the program can be tested in different means statement coverage, branch coverage, and path coverage are instances of these types of coverage. [6]. The strongest coverage criterion in structural testing is path coverage, it comprises of all three structural testing criteria, sometimes it also named as basis path testing.[6].

Test data generation is a central objective in software testing [7]. It is an efficient and effective way to generate equitable test data. The non-linear pattern of test data makes it more complex to generate optimal test data. The difficulty of generating test data is in tight loop with the level of problem, it increases or decrease with the involvedness of problem. Test data can be generated by adopting either manual procedure or through automated procedure, manually generation of test data required more efforts in comparison towards the generation of test data automatically for execution of test cases. Automatic, test cases (data) generation is key to find the adequate solution of problem of any size [8] [9].

The generation of test data is classified as an undecidability problem since it can be non-deterministic, making it an NP-hard problem, or there may be the possibility of infeasibility of existing outcome [10]. The program's exceptionally non-linear design makes it difficult for search algorithms to generate efficient and optimal test data from a non-linear, complex, and discontinuous input in the search space.

Path testing is a structural testing approach that ensures the execution of individual path at least once. The main issue with path testing is that how do we produce the effective test records that covers entire structure of the program in limited time period [1]. As it is not feasible to cover entire structure of the program, the path test method involves adopting subset of paths and searching of test data to unfold it. Many researchers have proposed number of methods for automatic generation of test data set for path testing [11] such as random testing approach , symbolic testing , dynamic testing and search based testing. All three approaches, to test data generation are inadequate to sustain enough appropriate test data. As a result, search-based testing is the trend of the day for generating test data [12].

With the gaining popularity for search-based testing many researchers start working on it, meta-heuristics search-based algorithms were considered stronger in this field because of its fault revealing capability. Genetic algorithm(GA), ant colony optimization (ACO) and simulated annealing (SA) are the popular meta-heuristic search-based algorithm [13]. But search-based algorithms still have some issues such as they may stuck in local optima, complete coverage, number of generations and execution time.

Despite search-based algorithm, artificial immune algorithms are also used for test data generation which shows significant improvement on searched based algorithm [14] [15]. Negative Selection algorithm [16] [17] , Colonel Selection algorithm [18] has applied in the field of test data generation. A hybrid approach based on artificial immune algorithm NSA and meta-heuristic algorithm PSO is also proposed for test data generation [19], which shows considerable advancement on meta-heuristic and artificial immune algorithms. Most of the work in the field of test data generation is proposed and implemented on modular/structural programming, only few researchers' works on object-oriented concepts.

This research proposed a new hybrid approach cantered on ant colony optimization and negative selection algorithm applications, to generate test data automatically on object-oriented system which have not been applied earlier and the results gives an effective test data that could traverse all program paths timely, when compared with the other techniques. The key aspects are summarised as follows:

- Applications of Negative Selection Algorithm such as (hamming distance) has been combined with in the functionality of ACO algorithm to achieve complete path coverage for test data generation.
- Path coverage is calculated using a fitness function that considers the reachability of each path.

- To validate the effectiveness of the proposed algorithm, some well-known programmes were used, as well as a comparison with random testing.

2. TEST DATA GENERATION

Generation of test data is a complex problem when used as automated. In earlier work search-based test data has widely been studied and most of the study was based on procedure-oriented system in structural testing. Various methods have been deployed in the literature for automatic generation of test data to enhance the coverage ratio and to bring the size of data down for different coverage criteria [20]. The mostly adopted test data generation techniques in which the researchers have the keen interest are random testing, symbolic testing, dynamic test data generation and search-based test data generation technique [6]. In random testing procedure, the test information has been chosen self-assertively from a search space, data generated through random technique shows high redundancy ratio, symbolic technique generated test data in static form and assign static values to variable instead of the real values and Dynamic techniques, necessitate the actual execution of source code for a limited input area [6].

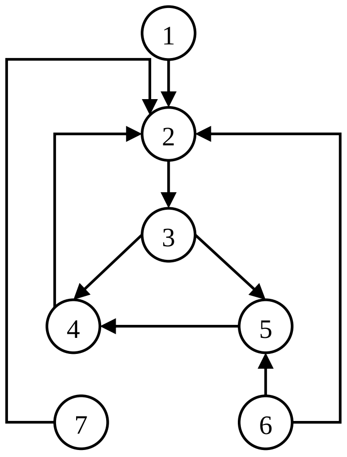

Fig.1. Control flow graph for Minmax

All the above three techniques are not so competent, they mostly generate the test data with high redundancy and the size of data always surge in this technique for slightly complex data structures and the size of input data is also inadequate. Search based test data generation is the most powerful searching techniques to locate the test data in the search space [21] [22]. SBST techniques uses search-based optimization algorithm alongside fitness function gets popularity in the research areas [23]. The general structure of SBST is presented in Fig.1 in a form of control flow grap [24]. To use the meta-heuristic approach to identify data in search space, we must first convert the source code into a control flow graph (CFG), that is a diagrammatic display of the source code [25].

Control flow graph (CFG) is a directed graph with the following definition:

$$F(G) = (N, E, s, e)$$

where, $N = N$ is a set of nodes in which each node corresponds to a statement; $E = E$ is a set of edges, each of which represents a control flow between nodes and is labelled with a predicate; S = Entry node; E = Exit node.

The control flow graph serves as a reference for locating an input that directs the software through various paths. CFG can be thought of as an optimization problem with the goal of increasing coverage criteria. Control flow graph for Minmax is given in Fig.1.

3. RELATED WORK ON TEST DATA GENERATION

Some studies on meta-heuristic algorithms like ACO, PSO, ABC, GA, FA, and artificial immune algorithms like NSA and clonal selection for the development of test data/cases have been published in recent years. The authors in [26] projected a new approach bases on PSO, in which the weight of inertia is modified based on fitness value. It uses branch Coverage as fitness criteria. Sanjay Singhal, [27] projected a hybrid approach by combining GA and PSO (GPSCA). It uses data flow coverage by applying dominance concept between two nodes and multi-objective coverage criteria. The authors in [28] projected an approach IGA based on Genetic Algorithm (GA) for automatic test case generation. They have done comparison of IGA with traditional GA for triangle classification problem using branch fitness criterion. The authors in [29] projected a GA based test data generator using multi path fitness. The approach can synthesize multiple test data to cover multiple target paths. Soma [30] projected an approach by combining the functionalities of scouts, employed and onlooker bees in ABC algorithm. The authors in [31] proposed a static based symbolic execution approach using ABC algorithm with branch distance as objective function.

The authors in [32] proposed a regression augmentation testing approach based on ABC algorithm with branch distance as objective function. Yang [33] proposed an approach based on ant colony optimization in which they have improved local pheromone strategy, pheromone volatilization co-efficient and global path pheromone with statement coverage, branch coverage and condition coverage as fitness value. Mao [34] proposed an approach in which they reformed ACO into discrete version by redefining local transfer, global transfer and pheromone update rule with customize branch fitness function. Sharma [35] has proposed an approach for automated software testing using meta heuristic technique based on improved ant algorithm in which she used statement, branch and modified decision/coverage as an objective function. Srivatsava [36] proposed a meta-heuristic technique based on ACO for state transition testing. Sayyari [37] has proposed an ACO and model-based testing approach. They have used Markov model for the re-formation of ACO. Aldeen [38] have proposed a new approach based on artificial immune system in which they have use the application of negative selection algorithm. Aldeen et.al. [19] projected a new approach based on NSA and GA for automated test data generation, the experimentation of the projected approach has been done on 11 real world programs, the projected approach is also compared with random testing approach and negative selection algorithm. Pachauri[18] has projected test data generation approach based on Clonal selection algorithm. They have used AI and NBD Approximation level with normalized branch distance as objective function to validate the test data.

Saini [39] has also projected an approach based on Clonal Selection algorithm. They have used Korel Distance function for branch predicate as objective function to validate the test data. The central objective of the above sited research is to explore the search capabilities of Meta–heuristic procedures such as ACO, PSO, ABC and GA and Artificial Immune algorithm NSA and Clonal selection algorithm on benchmark problems in software test data creation, comprising of classification of triangle, prime number generation, quadratic equation, largest number, telephone system, max-min etc. Meta-heuristic methods such as ACO, PSO, ABC and GA has excellent search capabilities, but all these algorithms have somehow lag in complete coverage and somehow stuck in local optima [40]. The Artificial Immune algorithm NSA and clonal selection are new approach in generation of test data. An immune algorithm has significant impact on the quality and coverage capabilities of test data generation and overcome the issues related with local optima [41].

4. BRIEF EXPLANATION OF ANT COLONY OPTIMIZATION AND NEGATIVE SELECTION ALGORITHM

4.1 ANT COLONY OPTIMIZATION

Marco Dorigo introduces the ACO algorithm by studying the foraging behaviour of the ant colony[10], [42]–[44]. Ant secretes pheromone on its way to share information with other ants during the foraging period. As every ant can perceive the trail of the pheromone, the forward direction can be regulated according to the pheromone's intensity on the route. Eventually, through many revisions, it can approach the food destination, with rapid speed and positive feedback process, ACO algorithm can identify the optimal solution. ACO has already been applied to solve the complex problems of optimization in different areas. However, ACO-based software testing has not been thoroughly investigated and remains a challenging subject [10] [45].

In an ant colony system (ACS), searching of optimal path is a process of generating solutions that can be referred to as a path on the construction graph $G = (V,E)$. The set of solutions can be linked to either the graph G node set V or the graph G edge set E [10] . The quantity of pheromone trail associated with edge (i,j) is supposed to indicate the learnt desirability of selecting node j when the ant is on node i and m ants are utilised to build a tour in the network given a graph with n nodes. If the kth ant is still on node i the current position (i.e., the node i set of neighbourhood nodes for such an ant) can be written as $N_{k(i)}{\cdot}N_{k(i)}$. This contains the nodes that ant i may visit in the next phase. In general, the selection of a node from $N_{k(i)}$ is done probabilistically at each step.

$$p_k(i,j) = \frac{\tau(i,j)\cdot[\eta(i,j)]^{\beta}}{\sum_{u\in N_{k(i)}} \tau(i,u)\cdot[\eta(i,u)]^{\beta}} \tag{1}$$

Once all ants have finished their tour, the pheromone on all edges is updated using the equation below. The goal of pheromone updating is to raise pheromone values associated with good or promising solutions while lowering those associated with negative ones.

$$\tau(i,j) \leftarrow (1-\alpha).\tau(i,j) + \Delta\tau(i,j) \tag{2}$$

The pheromone decay parameter $\alpha \in (0,1)$ is used in Eq.(2):

$$\Delta\,\tau(i,j) = \sum_{k=1}^{m}\Delta\tau_k(i,j) \text{ and } \Delta\tau_k(i,j) \tag{3}$$

Ant k has deposited a certain amount of pheromone on edge (i,j) [34]. It is commonly defined as:

$$\Delta\tau_k(i,j) = \begin{cases} \frac{1}{L_k} & if\ (i,j)\in T_k \\ 0 & Otherwise \end{cases} \tag{4}$$

where, T_k denotes the route taken by ant k, while L_k denotes the duration of the tour. It is clear from the definition of $\Delta\tau_k(i,j)$, that its value is greatly dependent on how well the ant has performed; the shorter the tour, the more pheromone is deposited.

$$\Delta\tau(i,j) = \begin{cases} \frac{1}{L_{gb}} & if\ (i,j)\in global-best-tour \\ 0 & Otherwise \end{cases} \tag{5}$$

In Dorigo's modified ant colony system (ACS), [46], $\Delta\tau(i,j)$ is based on only the best ant in the tour, where L_{gb} is the length of the best tour from the start of the trial [34].

4.2 NEGATIVE SELECTION ALGORITHM (NSA)

In an Artificial Immune System, the Negative Selection Algorithm (NSA) is possibly the most important strategy (AIS) [47]. The NSA is a self/nonself discrimination computational model that was first devised as a change detection tool. It is one of the initial AIS algorithms, and it is been employed in a variety of real-world applications [48]. The organic behaviour of the Natural Immune System (NIS), which is a compound organic organisation that uses rapid and dynamic methods to protect the body against predefined unfamiliar bodies called antigens, triggered AIS.

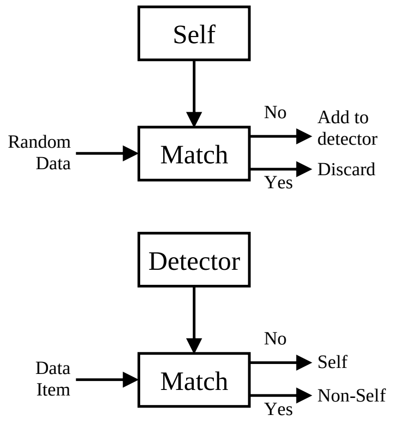

Fig.2. Negative Selection Algorithm

AIS's are a few of algorithms inspired by biologic systems, such as evolutionary algorithms, swarm intelligence, and neural networks that have sparked the interest of many researchers [49]. Its aim is to design immune-based algorithms for solving complex computations. One of the immune system's jobs is to recognise and classify all cells in the body as self or non-self. Negative selection is used to ensure that self-cells are accepted [50]. The primary idea underlying NSA is to create as many detectors as possible in the search area, and then utilise these detectors to determine whether new data is self or non-self [51]. The NSA is divided into two stages: generation (also known as training) and detection (also called testing stage). In the generation stage, a random method is utilised to generate the detectors, and the process is monitored. After the matched candidates are rejected, the leftovers are kept as detectors [17] [38]. The generation stage is accomplished when there are enough detectors (detector sets) formed [52]-[55]. In the detection stage, the detector sets generated in the previous stage are utilised to identify whether the input samples are self or non-self-samples. [56]-[59]. The Fig.2 describes the working of negative selection algorithm.

5. FRAMEWORK OF PROPOSED METHODOLOGY

In the proposed methodology, test procedures and both techniques must work in aligned manner to produce optimal outcome. The Fig.3 shows the process of test data generation. The following steps are required to generate the test data.

Step 1: Convert Program under test to control flow graph (CFG)

Step 2: Apply ACO to CFG for tracing the optimal path

Step 3: Local search is performed to update the pheromone trial alongside global best solution i.e., global search (if required)

Step 4: The application of negative selection is applied to reduce the redundancy and to minimize the size of the data.

Step 5: Path based fitness is computed to find the best solution.

Step 6: The fitness function's value can be utilised to guide technique in the next iteration.

Step 7: Identify traces and used them to count the information about coverage.

6. PROPOSED APPROACH

The primary objective of the ACO and NSA algorithm is to solve computational problems. We propose a hybrid strategy based on Ant Colony Optimization (ACO) and the application of the Negative Selection Algorithm (NSA) in this work. So that it can produce a high-coverage test data set with significant efficacy. The following is a formal definition of the test data creation problem, by combining the applications of both ACO and NSA technologies. Let a programme under test P to have a test data set as input i.e., $X=(x_1, x_2,\dots x_n)$, In the proposed approach, this can be treated as an ant's position vector. Assume that each input variable x_i, takes its values in the search space $D_i \in (1\leq i\leq n)$. As a result, the entire program's corresponding input domain can be represented as $D = D_1, D_2\dots.D_n$. It should create a test data set that traverses all elements in connection to a defined coverage criterion C. We use path coverage as a coverage criterion in our work. As a result, the objective of test data generation is to prepare a test input set TIS $=\{X\}$ that meets the highest possible path coverage criterion.

The search domain in traditional algorithm was a topology structure graph. An ant's neighbour region is a set of nodes that are adjacent to its current location in a graph. The position of each ant can be considered of as a test case in the test data generation process, and it is usually represented as a vector in the input domain. In this case, the domain is continuous

Euclidean space. Initially, m ants are placed randomly over the search domain. For every ant $k(1\leq k\leq m)$, Its position can be stated as $X_k=(x_{k1}, x_{k2},\dots x_{kn})$, The neighbour area can then be defined as a continuous region in which the distance between any point and ant k is less than or equal to a given constant r, where $X=(y_1, y_2,\dots y_n)$. In our algorithm, we use the Triangle classifier type example to represent the structure of an ant and its neighbours, The Triangle Type program has three input variables, if each input has a range of 0 to 9, the associated test case might be like: (1,1,1) that is the equilateral triangle and test suite may be TS = {{2,3,4,"Scalene triangle"}, {4,4,3,"Isosceles triangle"}, {3, 3, 3,"Equilateral triangle"}, { 1, 2, 3,"It is not a triangle"}, {2,1,0,"It is not a triangle"}, {1,2,0,"It is not a triangle"}, {5,3,5,"Isosceles triangle"}, {4,6,6," Isosceles triangle"}} .

The proposed hybrid approach slightly modified the pheromone update rule for test data generation. There is no specified linkage between the adjacent ants is described in the search space, for the same pheromone of individual ant is specified as $k(1\leq k\leq m)$, its pheromone can be represented as $\tau(k)$, Meanwhile, we have set 1 as a default value of (τ_0).

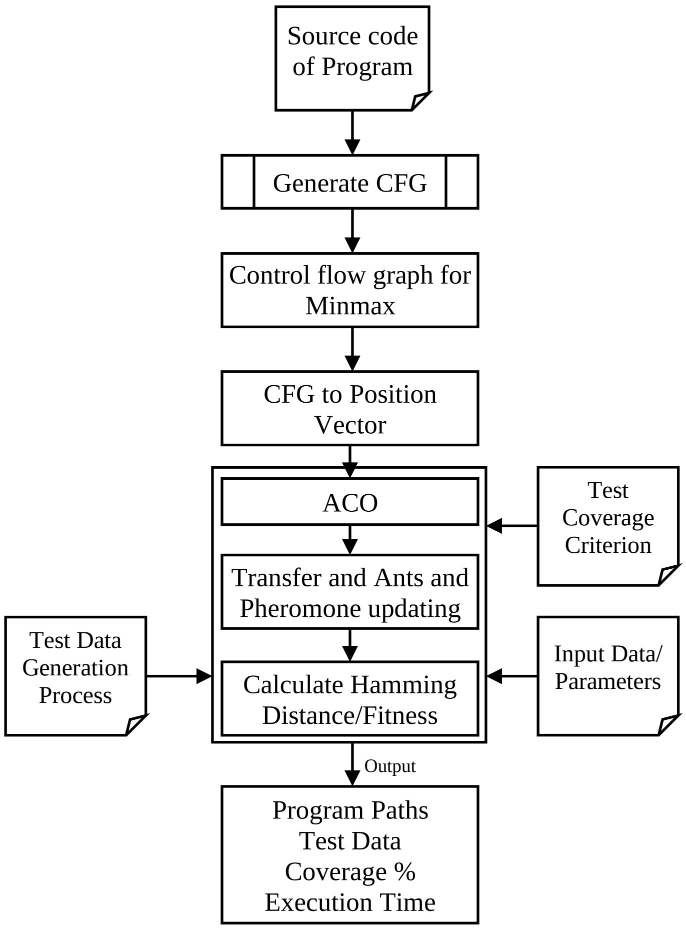
Source code of Program
Generate CFG
Control flow graph for Minmax
CFG to Position Vector
ACO
Transfer and Ants and Pheromone updating
Calculate Hamming Distance/Fitness
Test Coverage Criterion
Test Data Generation Process
Input Data/ Parameters
Output
Program Paths Test Data Coverage % Execution Time

Fig.3. Hybrid Test Data Generation Framework

6.1 LOCAL SEARCH AND GLOBAL SEARCH

During the scan, each ant seeks for a better solution in its immediate area. The local search is aligned with the shifting of ant positions. Its purpose is for each ant to randomly travel the solution in the proximity of the maximum radius r_{max}. Generally, we set the initial value of parameter r_{max} to a constant based on the characteristics of the problem. However, as the number of iteration times in searching increases, it will eventually decrease the value of r_{max}. Local shifting of ant can be well-defined, when ant k walks to a new neighbour position X_k and if $X_k>f(X_k)$, then ant can be transferred to new position. Otherwise, it will have to remain in its existing place. Here $f(X_k)$ is the fitness value of solution X_k.. Global search is applied when fitness of any node has higher value than the average fitness. i.e $f(X_k)>f_{avg}(X_k)$ in that case hamming distance is computed among the nodes to attain the global best solution

6.2 HAMMING DISTANCE

The application of Negative Selection Algorithm is used in the next step of the proposed strategy. After finding the new test data sets through Ant Colony Optimization, NSA is applied on those data sets, NSA not only identify the replication of test data generated through ACO but also support for complete path coverage and reduces the size of the test suites to elevate the performance and speed of the algorithm. Let's consider test data T_d, if it already exists in the newly generated test data set, discard it, from the set T_d, Otherwise, determine the hamming distance between the new detector T_{d1} from test data set T_d and all detectors T_{di} in the set and the smallest distance obtain will be compared with a threshold value. If the distance is lesser than the threshold value, then the test data will be removed from the test data set T_d, or else it is included to the refined set of test data, this approach aids in the coverage of the search area as far as possible., and it could cover more paths with a smaller amount of test data for the program under test. Subsequently go for the nearest test data from the set i.e., T_{d2} and calculate the fitness value of the new detector T_{d1} and T_{d2}, if the fitness value of $T_{d1}>T_{d2}$, interchange the test data T_{d2} with the test data T_{d1}. Following method is used to find the distance between test data.

1. Generate a new test data x, where $x \in S$;
2. Calculate the similarity of x with every test data d_i in $D\forall d_i \in D$ which is represent the hamming distance and could be calculating:

$$f_{aff}\left(d_i,x\right)=\sum_{i=0}^{n}\left(\overline{d_i \oplus x}\right) \tag{6}$$

6.3 FITNESS FUNCTION

The fitness function has significant impact on the validity of test data. Fitness function preferably applies for the refinement of test cases. In this study we have used path-based coverage criterion to validate the fitness of the code. Path based fitness can be calculated as:

$$PB_{Fitness} = 1-|\alpha \wedge \beta|/|\alpha \cup \beta| \tag{7}$$

where α and β are set of nodes in the targeted and executed paths, respectively $|\alpha \wedge \beta|$ presents the number of paired nodes in appropriate sequence between α and β. The path-based fitness for Minmax CFG of Fig.2 is 1-(3/6) = 0.5 because the node in target path set (α) contain nodes {1,2,3,4,5,6,7} and executed path set (β) contain nodes {1,2,3,5,6,7}, the fitness value is the ration between matched nodes in the correct order {1,2,3} and the number of nodes in the targeted path {1,2,3,5,6,7}. The Fig.5 depicts the proposed approach's flow chart, with TDGAN as the algorithm.

Algorithm: TDGAN

Input: Source code of program under test P, and its input variable list $X=(x_1, x_2,\dots x_n)$ where $\forall x \in S$; path testing coverage criterion C; The control flow graph CFG of program P; algorithm parameters α, φ, ρ_0, q_0, T, m and r_{max}; maximum evolution generation max*Gen*.

Output: Set of test data $D = (D_1, D_2 \dots D_n)$ this met the path coverage requirement. The set of paths that has been generated i.e. $U = (u_1, u_2 \dots u_n)$;

```
Initialization
if $x \exists S$
  goto initialization;
else
  $x \nexists$ in search space;
end if
Initialization:
for $k \to 1:m$ do
  for $i \to 1:n$ do
    Initialize the ith dimension (ant[$k$].$x$[$i$]) of position vector for
    the ant $k$;
  End For
  Calculate the fitness ant[$k$]·fitness of ant $k$;
  ant[$k$]·$\tau_0$=1, ant[$k$]·$count$=0;
  for $u \to 1$:m do
    ant[$k$]·record[$u$]=0;
  End For
End For
get the best one($g_{best}$)from $ant's_{fitness}$;
while gen<maxGen or $T_S$ does not reach full coverage of
criterion $C$ do;
for $k \to 1:m$ do
  Generate initial test data set randomly (candidate population);
  Test if the initial population reach to full coverage of path $U$
  goto end
  Generate a new test data $x$, where $x \in S$;
  Calculate the similarity of $x$ with every test data $d_i$ in $D \forall d_i \in D$
  by hamming distance and could be calculating as in Eq.(6).
  Check the distance $f_{aff}(d_i,x)$;
  if $f_{aff}(d_i,x) < \tau$
    then remove the new data set $x$;
  else
    add $x$ to $D$;
  end if
Repeat steps 25 to 29 until detector number>max or $D$ reach to
full coverage of paths $U$;
End
for $k \to 1:m$ do
  Update pheromone
  for $u \rightarrow 1$:ant[$k$]·count do
    ant[$k$]·record[$u$]=0;
  end for
    ant[$k$]·count=0;
end for
for $k \rightarrow 1:m$ do
  decode position ant[$k$].$x$[1….$n$] into a test case $tc_k \in T_S$;
  collect coverage information by executing program with $tc_k$
End For
End while
return $T_S$
```

7. EXPERIMENTAL EVALUATION

A comparison of real-world benchmark programs from the literature has been made to determine the performance of the proposed technique. These benchmark programs have been extensively applied in search-based testing by researchers. The program codes are being written in object-oriented programming languages such as Java, all these programs are designed by using the complex programming structure syntax such as relational operators, logical operator's conditional statement, control statements, modularity, and structure of classes etc. This made these programs suitable for analysing a variety of test data generation techniques. These programs often provide a complex data structure with various data types, such as integers, floats, characters, and strings. The Table.1 represent a summary of each program with the different number of arguments such as the number of variables in each program, the number of instructions, number of branches, lines in the code and complexity of the source code. The Table.1 shows different metrics used for program evaluation and its source.

Table 1. Benchmark Programs

Program	**Triangle Type**	**DayFinder**	**MinMax**	**Isprime**
Arguments	3	3	1:N	1
Instructions	50	168	83	34
Branches	16	24	6	6
Lines	13	24	12	11
Complexity	9	16	4	4

To prove that the ACO-NSA based test data generation approach is effective or not, followings test metrics are considered while evaluating the code such as:

- Average Coverage (ACG) i.e., the average of all test input of path coverage throughout multiple runs.
- Average Time (AT) i.e., the average execution time for all paths in seconds.

Different number of tests have been done for the above metrics such as for ACG and AT the value of test has been set to 1000 The experimental findings of two different approaches i.e. random testing and proposed novel algorithms are presented in response to four programmes in Table.3 and Table.4. The findings show that result of novel approach is better than those of Random testing for maximum number of programs. The novel ACO-NSA approach shows full coverage in maximum number of experiments done. The experimental setup of program triangle type in presented in Table.2. The Control Flow Graph of the program triangle type is presented in Fig.4 and the data in the table 2 represent the traced paths, complexity, input, and output.

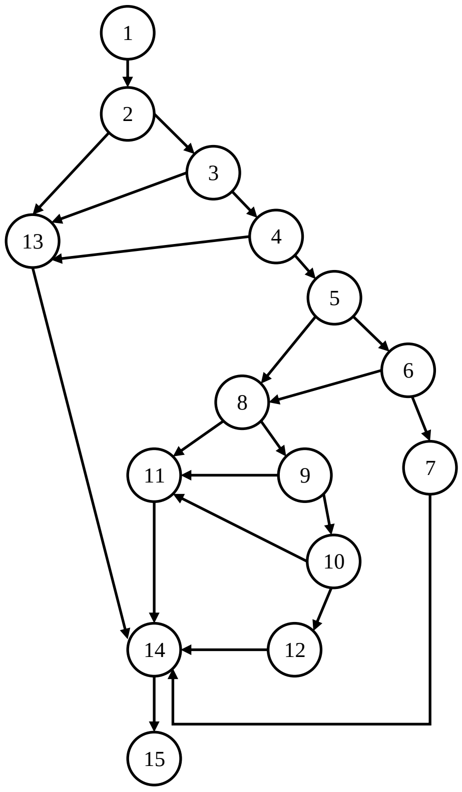

Fig.4. Control Flow Graph of Triangle Type

The Table.2 shows the flow of input data through different paths in triangle type program

Table 2. Path Covers by different input for Triangle Type program

Paths	**Input**	**Output**
1→2→3→4→5→12→14→15	2,3,4	Scalene
1→2→3→4→5→6→8→11→-14→15	4,4,3	Isosceles
1→2→3→4→5→6→7→14→15	3,3,3	Equilateral
1→2→13→14→15	1,2,3	Not a triangle
1→2→3→13→14→15	2,1,0	Not a triangle
1→2→3→4→13→14→15	1,2,0	Not a triangle
1→2→3→4→5→6→8→9→11→14→15	5,3,5	Isosceles
1→2→3→4→5→8→9→10→11→14→15	4,6,6	Isosceles
1→2→3→4→5→8→11→14→15	8,8,7	Isosceles

Table.3. Comparison analysis of metric average coverage (AC)

Program/Technique	**Random**	**Hybrid**
TriangleType	65.16	100
DayFinder	74.83	95.55
MinMax	63.7	100
Isprime	67.1	100

Table.4. Comparison analysis of metric average time (AT)

Program/Technique	Random	Hybrid
TriangleType	0.097	0.047
DayFinder	0.172	0.095
MinMax	0.108	0.025
Isprime	0.116	0.031

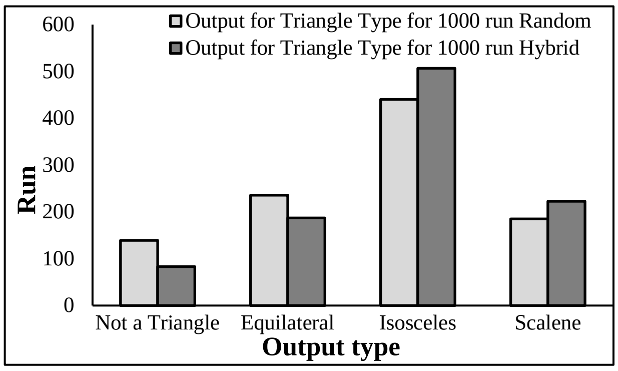

Fig.5. Test data coverage for Triangle Type

The Table.3 shows the comparison of metric average coverage. The Table.4 shows the comparison of metric average time. The Fig.5 shows the output of triangle type program for 1000 run.

8. DISCUSSION

This section of the paper presents the results of the experiments conducted to evaluate the performance of the proposed method i.e., novel approach based on ACO-NSA test data generation for path coverage. In the start the source code of the program is converted into control flow graph, then ACO-NSA is applied to generated automated test data. The results shows that the proposed approach generate least amount of test data in limited generations and have high coverage ratio. The results are compared with random testing to evaluate the performance of the projected approach. The performance is measure in terms of Average Coverage (ACG) and average time (AT). This section of the paper presents the performance of the proposed approach for different benchmark programs which have been the pivot point for the researchers in the field of search-based test data generation and in immune algorithms.

All the benchmark programs are widely applied by the researchers for test data generation. These benchmark programs design flow structure made them suited for testing a variety test data generation technique. All such programs have different data structures, line of codes (LOC), arithmetic, relational and logical operators, loops and nested loops, conditional statements, arrays, functions and classes and complexity levels. The Table.1 gives a brief description of each program.

The studies were carried out in a Microsoft Windows 10 environment with an Intel Core TM i7 2.10 GHz 64-bit processor and 8 GB RAM. The eclipse 20-3 Java platform is used to code the program, the MATLAB platform is used to code the method's implementation and the generated test data is verified using testing tool TestNG and coverage is recorded through tool ECL Emma.

This section depicts the outcomes of each program. The "triangle type classifier (Tritype)" is highly recognized programming application for testing. It seems to be a simple application for testing process, but it has all the requirements suitable for testing such as data structures, conditional and logical operators, conditional and logical statements, functions, and arrays. It takes three input variables and uses them to decide the triangle type (scalene, isosceles, equilateral, and not a triangle). The size of the search space is proportional to the data type if it is assumed to be integer of type, it may consume two bytes of memory for individual variable declared. It will be very difficult to design test cases corresponding to such large range of data from the appropriate domain corresponding to the data type of the variable. The proposed approach guided the method to generate appropriate test data from the domain to achieve the full path coverage. Path fitness has been applied along with the proposed method to achieve the quality data. The probability of finding the accurate value for all three variables that execute the required path such as isosceles triangle depends upon the three variables i.e., the probability of having any type of triangle will be $1/3^{rd}$ of all. The analysis reveals that the novel ACO-NSA is more efficient than random testing for the triangle type classifier program, isprime, dayfinder and minmax for test data generation. The number of generations required in novel ACO-NSA is comparatively very less as compared to random testing. It can be concluded from the results in (Table.3 and Table.4) that the proposed novel ACO-NSA approach is suitable for use in programs that have complex path with loops and nested selection because it can accomplish comprehensive path coverage.

9. CONCLUSION

This paper proposed a novel approach based on ACO-NSA, a hybrid approach that incorporates ACO and NSA for the creation of automated software test data. This technique applied path-based fitness functions to modify random detector generation, to produce optimized and minimal quantity of detectors (test data set) and guide the search of test data to paths with minimal probability of being executed. The proposed approach increases the percentage of path coverage while avoiding redundant data and enhances reliability and effectiveness of test data generation. The newly generated results show the significant improvement in path coverage, including in complex paths. The average coverage (ACG) also improved significantly in the novel ACO-NSA approach, the approach also has high success rate with low execution time and get a smaller number of generations to execute the source code. The proposed approach yields better results by reducing, the amount of generated test data while reducing the number of generations.

REFERENCES

[1] S.C. Ntafos, "A Comparison of Some Structural Testing Strategies", *IEEE Transactions on Software Engineering*, Vol. 14, No. 6, pp. 868-874, 1988.

[2] G.D. Everett and R. McLeod, "*Software Testing: Testing Across the Entire Software Development Life Cycle*", Wiley, 2006.

[3] K. Sneha and G.M. Malle, "Assistant Professor in Computer Science Department", *Proceedings of International*

Conference on Energy, Communication Data Analysis, pp. 77-81, 2017.

[4] M.A. Jamil, M. Arif, N. Sham, A. Abubakar and A. Ahmad, "Software Testing Techniques : A Literature Review", *Proceedings of International Conference on Information and Communication Technology*, pp. 1-6, 2016.

[5] N. Anwar and S. Kar, "Review Paper on Various Software Testing Techniques and Strategies", *Global Journal of Computer Science and Technology: C Software and Data Engineering*, Vol. 19, No. 2, pp. 1-8, 2019.

[6] 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.

[7] V. Garousi and M.V. Mantyla, "A Systematic Literature Review of Literature Reviews in Software Testing", *Information and Software Technology*, Vol. 80, pp. 1339-1351, 2016.

[8] S. Parnami, "Testing Target Path by Automatic Generation of Test Data using Genetic Algorithm", *International Journal of Information and Computation Technology*, Vol. 3, No. 8, pp. 825-832, 2013.

[9] K. Lakhotia and P. Mcminn, "Automated Test Data Generation for Coverage : Haven't We Solved This Problem Yet ?", *Proceedings of International Conference on Practice and Research Techniques*, pp. 1-6, 2009.

[10] M. Dorigo, M. Birattari and T. Stützle, "Ant Colony Optimization Artificial Ants as a Computational Intelligence Technique", *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4, pp. 28-39, 2006.

[11] S. Anand, "An Orchestrated Survey of Methodologies for Automated Software Test Case Generation Orchestrators and Editors", *The Journal of Systems and Software*, Vol. 86, No. 2013, pp. 1978-2001, 2015.

[12] M. Harman, S.A. Mansouri and Y. Zhang, "A Comprehensive Analysis and Review of Trends Techniques and Applications", *Search Based Software Engineering*, Vol. 12, pp. 1-18, 2009.

[13] M. Harman and P. Mcminn, "A Multi - Objective Approach To Search - Based Test Data Generation", *Proceedings of 9th Annual Conference on Genetic and Evolutionary Computation*, pp. 1098-1105, 2007.

[14] W. Rhmann, "Dynamic Test Data Generation using Negative Selection Algorithm and Equivalence Class Partitioning", *International Journal of Advanced Research in Computer Science*, Vol. 8, No. 3, pp. 189-192, 2017.

[15] J. Al-Enezi, M. Abbod and S. Alsharhan, "Artificial Immune Systems-Models, Algorithms and Applications", *International Journal of Research and Reviews in Applied Sciences*, Vol. 3, No. 3, pp. 118-131, 2010.

[16] R. Rahnamoun, "Distributed Black-Box Software Testing Using Negative Selection", *International Journal of Smart Electrical Engineering*, Vol. 2, No. 3, pp. 151-157, 2013.

[17] I. Journal, C. Vision, S. Mustafa, R. Mohamad and U. Teknologi, "Automated Path Testing using the Negative Selection Algorithm", *International Journal of Computational Vision and Robotics*, Vol. 7, No. 1-2, pp. 1-15, 2017.

[18] A. Pachauri, "Use of Clonal Selection Algorithm as Software Test Data Generation Technique", *Proceedings of International Conference on Advanced Computing and Communication Technologies*, Vol. 2, No. 2, pp. 1-5, 2012.

[19] S.M.M. Id, R. Mohamad and S. Deris, "Optimal Path Test Data Generation based on Hybrid Negative Selection Algorithm and Genetic Algorithm", *PLOS One*, Vol. 34, No. 3, pp. 1-21, 2020.

[20] S.M. Mohi-Aldeen, S. Deris and R. Mohamad, "Systematic Mapping Study in Automatic Test Case Generation", *Frontiers in Artificial Intelligence*, Vol. 265, pp. 703-720, 2014.

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

[22] G.I. Latiu, O.A. Cret and L. Vacariu, "Automatic Test Data Generation for Software Path Testing using Evolutionary Algorithms", *Proceedings of 3rd International Conference on Emerging Intelligence Data Web Technology*, pp. 1-8, 2012.

[23] M. Harman, P. Mcminn and R. Court, "A Theoretical and Empirical Analysis of Evolutionary Testing and Hill Climbing for Structural Test Data Generation", *Proceedings of International Symposium on Software Testing and Analysis*, pp. 73-83, 2007.

[24] Y. Chen, Y. Zhong, T. Shi and J. Liu, "Comparison of Two Fitness Functions for GA-based Path-Oriented Test Data Generation", *Proceedings of International Conference on Natural Computation*, pp. 1-15, 2009.

[25] H. Tahbildar and B. Kalita, "Automated Software Test Data Generation: Direction of Research", *International Journal of Computer Science and Engineering Survey*, Vol. 2, No. 1, pp. 1-12, 2011.

[26] X. Zhu, "Software Test Data Generation Automatically Based on Improved Adaptive Particle Swarm Optimizer", *Proceedings of International Conference on Computational and Information Sciences*, pp. 1300-1303, 2010.

[27] 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.

[28] D.A.N. Liu, X. Wang and J. Wang, "Automatic Test Case Generation based on Genetic Algorithm", *Proceedings of International Conference on Control Systems, Computing and Engineering*, Vol. 48, No. 1, pp. 411-416, 2013.

[29] M.A. Ahmed and I. Hermadi, "GA-based Multiple Paths Test Data Generator", *Computer and Operation Research*, Vol. 35, pp. 3107-3124, 2008.

[30] S. Sekhara, B. Lam, M.L.H. Prasad and S. Ch, "Automated Generation of Independent Paths and Test Suite Optimization using Artificial Bee Colony", *Procedia Engineering*, Vol.12, No. 1, pp. 1-5, 2021.

[31] S.S. Dahiya, J.K. Chhabra and S. Kumar, "Application of Artificial Bee Colony Algorithm to Software Testing", *Proceedings of International Conference on Software Engineering*, pp. 149-154, 2010.

[32] B. Suri, P. Kaur, D.B. Suri and P. Kaur, "Path Based Test Suite Augmentation using Artificial Bee Colony Algorithm", *International Journal for Research in Applied Science and Engineering Technology*, Vol. 2, No. 9, pp. 156-164, 2014.

[33] S. Yang, T. Man and J. Xu, "Improved Ant Algorithms for Software Testing Cases Generation", *The Scientific World Journal*, Vol. 2014, pp. 1-13, 2014.

[34] C. Mao, L. Xiao, X. Yu and J. Chen, "Adapting Ant Colony Optimization to Generate Test Data for Software Structural Testing", *Swarm Evolutionary Computing*, Vol. 20, pp. 23-36, 2015.

[35] P. Sharma, "Automated Software Testing using Metahurestic Technique Based on Improved Ant Algorithms for Software Testing", *Proceedings of International Symposium on Electronic System Design*, pp. 3505-3510, 2010.

[36] P.R. Srivastava, "Automated Software Testing using Metahurestic Technique Based on An Ant Colony Optimization", *Proceedings of International Conference on Advanced Computing*, pp. 1-13, 2010.

[37] F. Sayyari and S. Emadi, "Automated Generation of Software Testing Path based on Ant Colony", *Proceedings of International Conference on Technology, Communication and Knowledge*, pp. 11-12, 2015.

[38] S.M. Mohi-Aldeen, R. Mohamad and S. Deris, "Application of Negative Selection Algorithm (NSA) for Test Data Generation of Path Testing", *Applied Soft Computing*, Vol. 49, pp. 1118-1128, 2016.

[39] P. Saini and S. Tyagi, "Test Data Generation for Basis Path Testing using Genetic Algorithm and Clonal Selection Algorithm", *International Journal of Science and Research*, Vol. 3, No. 6, pp. 2012-2015, 2014.

[40] C. Mao, X. Yu, J. Chen and J. Chen, "Generating Test Data for Structural Testing Based on Ant Colony Optimization", *Proceedings of International Conference on Quality Software*, pp. 98-101, 2012.

[41] S.M. Mohialdeen, R. Mohamad and S. Deris, "Automatic Test Case Generation for Structural Testing using Negative Selection Algorithm", *Proceedings of International Conference on Recent Trends in Information and Communication Technologies*, pp. 1-12, 2014.

[42] A.E. Rizzoli, "Ant Colony Optimization for Real-World Vehicle Routing Problems", *Swarm Intelligence*, Vol. 133, No. 1, pp. 87-151, 2007.

[43] M. Dorigo, V. Maniezzo and A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man and Cybernetics-Part B*, Vol. 26, No. 1, pp. 1-26, 1999.

[44] K. Socha and M. Dorigo, "Ant Colony Optimization for Continuous Domains", *European Journal of Operational Research*, Vol. 185, No. 3, pp. 1155-1173, 2008.

[45] S. Nallaperuma, M. Wagner and F. Neumann, "Ant Colony Optimisation and the Traveling Salesperson Problem - Hardness, Features and Parameter Settings Categories and Subject Descriptors", *Proceedings of International Conference on Companion on Genetic and Evolutionary Computation*, 2013.

[46] C.S.G Dhas and T.D. Geleto, "D-PPSOK Clustering Algorithm with Data Sampling for Clustering Big Data Analysis", Academic Press, 2022.

[47] J. Timmis, A. Hone, T. Stibor and E. Clark, "Theoretical Advances in Artificial Immune Systems", *Theoretical Computer Science*, Vol. 403, No. 1, pp. 11-32, 2008.

[48] S. Stepney, "Conceptual Frameworks for Artificial Immune System", *International Journal of Unconventional Computing*, Vol. 1, No. 3, pp. 315-338, 2005.

[49] D. Dasgupta, "Advances in Artificial Immune Systems", *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4, pp. 40-43, 2006.

[50] M. Ponnusamy, P. Bedi and T. Suresh, "Design and Analysis of Text Document Clustering using SALP Swarm Algorithm", *The Journal of Supercomputing*, Vol. 12, pp. 1-17, 2022.

[51] Z. Liu, T.A.O. Li, J.I.N. Yang and T.A.O. Yang, "An Improved Negative Selection Algorithm Based on Subspace Density Seeking", *IEEE Access*, Vol. 5, pp. 12189-12198, 2017.

[52] H. Hou and G. Dozier, "An Evaluation of Negative Selection Algorithm with Constraint-Based Detectors", *Proceedings of 44th International Conference on Recent Trends in Information Technology*, pp. 134-139, 2006.

[53] P. Agarwal, "Nature-Inspired Algorithms: State-of-Art, Problems and Prospects", *International Journal of Computer Applications*, Vol. 100, No. 14, pp. 14-21, 2014.

[54] E. Alba and J.F. Chicano, "Software Testing with Evolutionary Strategies", *Lecture Notes in Computer Science*, pp. 50-65, 2006.

[55] I. Hermadi, C. Lokan and R. Sarker, "Dynamic Stopping Criteria for Search-Based Test Data Generation for Path Testing", *Information and Software Technology*, Vol. 56, No. 4, pp. 395-407, 2014.

[56] S. Kumar, D.K. Yadav and D.A. Khan, "Artificial Bee Colony based Test Data Generation for Data-Flow Testing", *Indian Journal on Science and Technology*, Vol. 9, No. 39, pp. 1-13, 2016.

[57] C.C. Michael, G. McGraw and M.A. Schatz, "Generating Software Test Data by Evolution", *IEEE Transactions on Software Engineering*, Vol. 27, No. 12, pp. 1085-1110, 2001.

[58] 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.

[59] R. Malhotra, "Comparison of Search based Techniques for Automated Test Data Generation", *International Journal of Computer Applications*, Vol. 95, No. 23, pp. 4-8, 2014.