Journal of Electronic Testing (2022) 38:623–636
https://doi.org/10.1007/s10836-022-06033-8

Smell Detection Agent Optimization Approach to Path Generation in Automated Software Testing

S. S. Vinod Chandra[1] · **S. Saju Sankar**[2] · **H. S. Anand**[3]

Received: 8 July 2022 / Accepted: 4 November 2022 / Published online: 2 December 2022

Abstract
Software testing is the most crucial stage in the software development process. Structural testing, functional testing and models that even support hybrid testing are different software testing techniques. Basic path testing, the most significant structural testing approach, is focused on evaluating software source code. The method emphasizes developing test data inputs to produce all feasible and efficient test paths that connect to all nodes and edges of the graph. The objective is to define the number of independent paths that can define the number of test cases needed to maximize test coverage. It ensured the execution of every statement and condition at least once. A nature-inspired Smell Detection Agent (SDA) algorithm is proposed in this paper to select all paths and prioritize the feasible solution. This algorithm is an optimization algorithm suitable for identifying optimal paths with priority. The concept is derived from the natural behaviour of canines that identified optimal path from source to the destination. The SDA algorithm is based on the evaporation of smell molecules in the form of gas and the perception capability of a smelling agent. The number of linearly independent paths through a programme module is measured by creating a Control Flow Graph of the code, which measures cyclomatic complexity. SDA algorithm gives significant increases in performance while considering the cyclomatic complexity. Complexity analysis of SDA trends to be in the O(E+V log V), while the competitor algorithms have an exponential growth of $O(n^2)$. Various experiments were also carried out to emphasis the relevance of the proposed method. Ten different benchmarked applications has been taken for experimental analysis and it was observed to have an increased path coverage of 8% when SDA was used over the traditional methods. Also, the time complexity was reduced by 22%, which shows the powerfulness of the proposed SDA algorithm.

Keywords SDA algorithm · Software testing · Structural testing · Basic path testing · Test case calculation

1 Introduction

The development of appropriate software products for specific tasks is necessary and involves many human and financial resources. The products thus developed should ensure high quality before deployment or release. Any type of product or process testing can be carried out to ensure quality. Similarly, software testing should also be carried out to verify and validate its components, mainly programs. Both these activities can be achieved by structural and functional testing of the software. Valid test cases are designed in both these methods and test data for the same is generated either manually or using automated methods [24]. As exhaustive testing of the software product is impractical due to time and cost constraints and as redundancy of the test cases also needs to be controlled, an effective optimization method needs to be employed.

Responsible Editor: V. D. Agrawal

✉ S. S. Vinod Chandra
vinod@keralauniversity.ac.in

S. Saju Sankar
sajusankars2019@gmail.com

H. S. Anand
anandhareendrans@mgits.ac.in

[1] Department of Computer Science, Thiruvananthapuram, University of Kerala, Thiruvananthapuram, India

[2] Department of Computer Engineering, Government Polytechnic College, Punalur, India

[3] Department of Computer Science and Engineering, Muthoot Institute of Technology and Science, Kochi, India

Designing an optimization algorithm for the automated generation of optimal and feasible test paths in structural testing is the major focus of this work. Various algorithm design strategies have been tried [10, 17, 19, 23], but the efficiency and accuracy obtained from the group of nature-inspired optimization algorithms were commendable for optimizing and automation of the generation of optimal and feasible paths in structural testing is explained in the work. SDA algorithm is implemented in the formulation of feasible paths and a comparison of the results obtained while using other nature-inspired optimization algorithms is also included in the study.

Nature Inspired Algorithms in Automated Software Engineering Designing smart and optimized algorithms is essential in the software development life cycle, especially during the software testing phase. The appropriate selection of test suites is a combinatorial problem that uses various nature-inspired algorithms for automatic test data generation in software testing.

Ant Colony Optimization (ACO) algorithm is one of the popular nature-inspired model that considers statement coverage, branch coverage, and condition coverage as the test adequacy criteria [14, 20]. Different ACO algorithms based on pheromone level intensity updating are extracted for the generation of test suites for both path coverage and its prioritization.

For the automatic generation of test cases, the Genetic Algorithm(GA) is one method. It uses real numbers instead of binary numbers to improve the basic Genetic algorithm. Experiments were conducted based on a search-based Genetic algorithm, and it uses a tool EVOSUITE for obtaining test cases necessary for data flow testing. Another method suggested a hybrid Genetic algorithm called HGA to generate test cases by evaluating a fitness function based on statement coverage [28]. The new mixed technique is built by a hill-climbing technique integrated with a simple genetic algorithm. A modified genetic algorithm called regenerate genetic algorithm solved the population ageing problem that provided better coverage in both path and branch for various issues.

There has been a lot of proposal on the various software testing process. Koteswara has proposed test path coverage for the shortest path. They have proposed a path coverage genetic algorithm. Test case testing and optimization are important areas and algorithms employed genetic algorithm and Artificial bee colony [18].

Particle Swarm Optimization (PSO) can be used to generate test suites needed for data flow testing. The observed results indicate that PSO performs much better than the Genetic algorithm [27]. The number of generations is less in PSO when compared to the Genetic algorithm. A modified PSO algorithm called Discrete Quantum PSO was also developed to generate test cases automatically.

Reduced Adaptive PSO is a modified algorithm used for automatic test data generation. The evolution equation is modified after removing the velocity component of the PSO algorithm, and the inertia weight is the factor considered. This new approach provides better convergence speed.

A hybrid algorithm called Adaptive PSO - Genetic algorithm removed the problem of immature convergence from PSO and the slow convergence of the genetic algorithm. Here, a new objective function combining dominance weight, branch weight and branch distance in a Control Flow Graph (CFG) of the program under test is suggested. This method's effectiveness is tested with other algorithms - Differential Evolution, PSO, Genetic algorithm and ACO [9]. This approach produced a better result than different algorithms. The PSO algorithm, called Accelerating PSO (APSO) for data flow testing and APSO, is a better model between exploration and exploitation [15].

Firefly Algorithm is a bioinspired meta-heuristic algorithm based on fireflies' flashing behaviour. Generating test cases in structural testing and functional testing is obtained by applying the Firefly algorithm. Chaotic firefly, a modified version of the firefly algorithm, is employed to generate test cases for white box testing.

A tool named 'Optimal Firefly Test Sequence Generator' was also developed for structural testing. The software is converted into a state machine diagram, and the algorithm is used to generate a prioritized test sequence for the state machine [12]. Various results in the case studies are taken and compared with the ACO algorithm. The results obtained using the Firefly algorithm are less redundant than the ACO algorithm [8].

Contribution Path refers to the flow of execution or sequence of commands and conditions in a definitive direction. Basis path testing finds all the possible executable paths of the code. In a coding environment, test cases are generated to test all possible workflow and to make sure the code works perfectly in all possible environments. When the application is big enough, the number of test cases that need to be created will also be high in number. An automated process of test case generation by employing a path generation technique is proposed in this work. This work is unique because the authors have deployed nature-inspired algorithms for generating test cases by automating the testing process. The smell detection agent algorithm is a nature-inspired algorithm developed from the olfactory sensing power of canines. Incorporating an SDA based algorithm for automatic software testing makes this paper unique from other publications in this area. This work is first of its kind in generating test paths by deploying smell detection agent algorithm.

The rest of the paper is planned as follows. The way in which nature inspired algorithm is used in automated software engineering is discussed in the first part of the manuscript. A generalised algorithm called smell detection algorithm is explained in detail with the methodology which is basically used in this work for automated test case generation. Complexity analysis and user test cases of two different code segments are also mentioned in the manuscript to claim the validity of the proposal.

2 Methodology of Agent Based Algorithms

All nature-inspired optimization algorithms are normally based on different parameters or observations, and these parameters decide the application of the algorithm. Smell detection or sniffing of canines helped find the shortest path from source to destination [13, 22]. This natural phenomenon of canines is simulated in the software testing process, in which optimal test paths are suitably found out.

Smell trails increase from source to destination and the shortest path is obtained without any redundancy. The phenomenon revolves around sniffing mode, trailing mode and random mode. The evaporation of smell molecules from the source and the movement of these molecules in the direction of the agent will be developed. The path is obtained from the source of the smell.

Since the dawn of time, man has been perplexed by nature.Several natural events have been used to develop optimisation algorithms in computational science [7]. Natural selection has resulted in the emergence of several emergent behavioural patterns in creatures with optimising overtones. These were detected separately and in nature-inspired algorithms.

Nature-inspired computing includes many optimization algorithms within each source of inspiration and approach for mapping it into the computational realm. Until now, most algorithms have primarily dealt with an agent or a group of agents that demonstrate collective intelligent behaviour [25, 26].

Indeed, even today, canines are utilized by wrong doing discovery powers to follow the path of lawbreakers from the location of the crime. A particularly regular interaction can be utilized to make answers for the chose enhancement issues. Numerous improvement issues can't be settled traditionally and are asymptotically NP-hard. In such settings, nature-enlivened marvels like creature conduct can be utilized to take care of computational issues. A calculation, which adjusts the regular cycle of sniffing or smell trail discovery, is the primary thought of the work. The issue to be tackled is viewed as a surface with smell trails and computational specialists motivated by canines to distinguish a streamlined way, which will comprise the arrangement [3, 11, 21].

Every biologically inspired optimization algorithm is usually based on several critical modeling parameters [2, 4, 5]. These parameters are usually decided through careful observation and understanding of the biological system on which the algorithm is inspired. To develop an algorithm inspired by the phenomenon of smell, the Brownian movement of the smell molecules towards the agent, the agent's training movement towards the molecule should be considered. Thus, the following mode for the algorithm can be modelled [1, 6].

1. The gaseous molecules of smell evaporate in the direction of the Smell Agent. This is termed the sniffing mode.
2. The smell agent trailing the part of the gaseous molecules smell and eventually identify it source. This is termed the trailing mode.
3. In a situation where the agent loss its trail during the search, a position is selected randomly and the agent move towards this position hoping to sniff the smell molecule again. This is termed the random mode.

2.1 Computational Model

The idea of the canine way following conduct can be plotted in organize math. The fundamental climate is a Cartesian rectangular plot with indicated furthest points characterizing the region. These qualities might be changed depending on the particular computational limitations of the issue. All focuses in the plot are not navigable but rather chose irregular focuses (inspecting of focuses) that the SDAs can visit. These focuses are called smell detects that aides in centring the critical thinking to a subset of focuses. Every one of these smell spots is put away by two boundaries. One is smell trail or worth from an objective or objective. Another is the mark worth of any SDA that has visited or denoted a smell spot. Figure 1 delineates this thought that comprises a source, objective, smell spots, and smell range for an SDA to travel [16].

There are two parameters in each SDA. One is their allotted signature esteem that can be utilized to stamp smell spots, and the other is the range esteem that shows their olfactory capacity. Diverse SDAs can have distinctive smell recognition capacities. This worth changes in a sweep of round region reviewed or sniffed by a specific SDA at a point. That is, each SDA can have distinctive smell identification capacity esteems. The smell spots are dispersed over the plotted region with various qualities. This may lead an SDA to sniff and reach a specific point. The above thought can be carried out as an information structure that is valuable in a calculation plan.

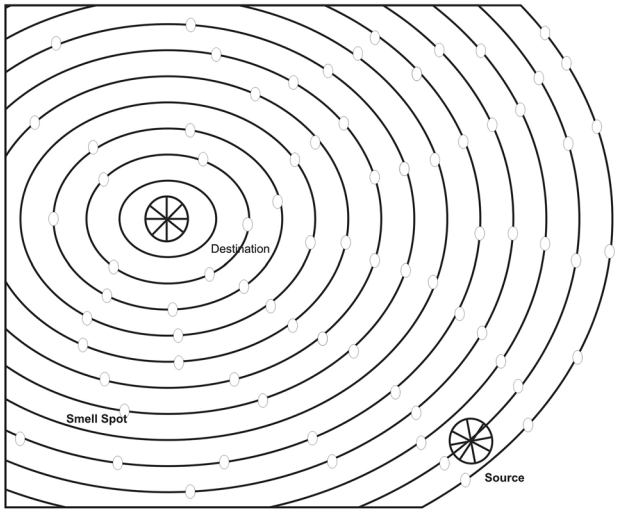

Fig. 1 Smell surface and smell spots

A Sensory Mechanism in Nature Nature has supplied organic entities with different tactile instruments for their endurance. Canines, particularly species like dogs, have uncommon olfactory cells, giving them a much prevalent ability to distinguish utilizing smell. Canines have profoundly tuned receptor destinations in their noses that can bolt on to this fragrance, accordingly recognizing the way embraced from the beginning of aroma. On the path, they additionally invigorate their olfactory cells by taking fast sniffs and afterwards purging their noses to give the olfactory cells a rest. Another inquisitive conduct of canines is that they pee in various spots to check an area. Different canines will distinguish it as an involved domain because of the solid and particular smell of the pee. These two eccentricities of canines have been utilized to make smell discovery specialists (SDA) recognize smell trails in the climate and mark the way they embrace.

Sniffing Mode In a practical situation, the agent should be able to sniff the smell particles and intuitively follow this particle with the hope of identifying its source. This usually bemuses the agent due to variation in the temperature and molecular mass of the smell particles, making part trailing a challenging task. While exploring the search space, the concentration of smell molecules may become higher than the current position of the agent, in this case, the agent moves towards this position. This way, the agent continues to trail the position of all molecules with higher concentration until the molecule with the overall best fitness (smell source) is identified. This behaviour of the agent would be model as the sniffing mode in smell agent optimization. The process of SDA is initiated by a randomly generated initial position (population) of smell molecules. The size of the population depends on the total number of molecules of smell evaporating from the smell source. Assuming the smell molecule is denoted as N and the hyperspace where the smell molecule is evaporating is denoted as D, then the population of the smell molecules will be assigned a position as to where i = {1, 2..., N}, t is the present position of a smell molecule. The position vector in the equation enables the agent to determine the region with the highest concentration in the search space. For example, consider the coordinate positions given in Fig. 2.

Each cycle in the cell depicts a molecule of smell. The number of columns in the figure represents the Dimension (D) or solution search space to be explored, while the number of rows represents the population of smell molecule exploiting the search space.

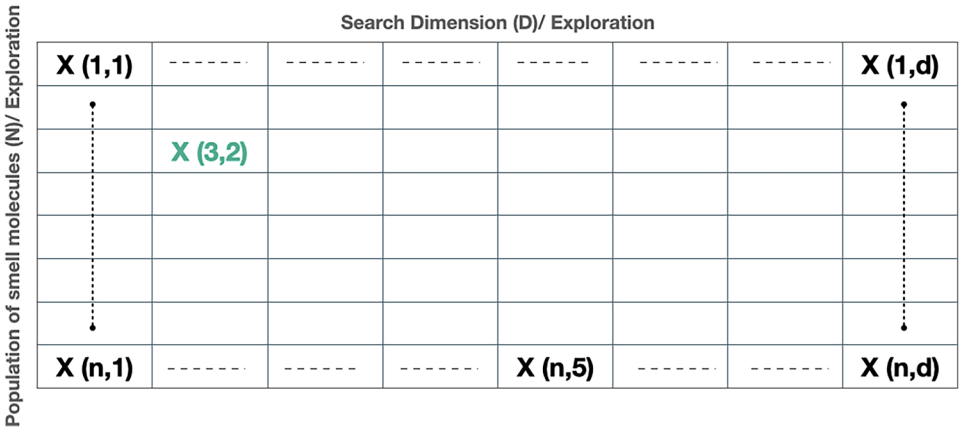

Fig. 2 Smell positions in Hyperspace

From Fig. 2, if N is 3 and the index of the molecule with the highest concentration $x_t(3,2)$, this indicates that the smell molecule with the highest concentration, in this case, is at the coordinate (3,2) of the entire search space indicated as a circle with green color. Since the smell molecule evaporates and travels through the air in the direction of the agent, each molecule can be said to maintain a uniform velocity in the direction of the agent, provided the intensity of the air medium is constant. This movement velocity of the smell molecule is denoted as V. The velocity vector V is the diffusion vector (which is a displacement of the smell molecule from the smell source/origin). Thus, the position of the smell molecules considering the velocity vector will be given as $X_i^{t+1} = X_i^t + V_i^t$

Since every smell molecule will their corresponding velocities for which they move and update their position in the search space. The preliminary theory of molecule movement and the heuristic derivation of the velocity distribution function will be used to develop the velocity update model.

Trailing Mode While exploring the search space, the concentration of smell molecules may become higher than the agent's current position. In this case, the agent moves towards this position. This way, the agent continues to trail all molecules with higher concentration until the molecule with the overall best fitness (smell source) is identified. In a practical situation, the agent should be able to sniff the smell particles and intuitively follow this particle with the hope of identifying its source. This usually confuses the agent due to variation in the smell particles' temperature and molecular mass, making part trailing a challenging task. Also, every agent has a specific olfaction capacity, depending on its size, olfactory lobe, and psychological and physical condition. For example, larger olfactory lobe size indicates stronger olfaction favouring exploitation, while smaller olfactory lobe size indicates poor olfaction, which indicates poor exploration. Since the proposed SDA is generalized for all agents, the agent's olfaction capacity will influence the SDA precision and convergence. This behaviour of the agent would be model as the trailing mode in smell agent optimization.

In Fig. 3 the agent is represented as a canine and the object evaporating the smell molecule is depicted in the grey circle. The dotted lines (labeled X) denote the direction of the evaporation of the smell molecules, while the thick line represents the path with a high concentration of smell (optimum path). The paths labeled X_{fe} are all feasible paths (path with lower smell concentration), which could lead the agent to the smell source if followed. However, the agent is usually restricted to follow only the optimum path all through the search process. At every stage in the searching process, the agent takes note of the X_{worst} position (feasible paths in a generation) and uses this information to restrict its movement within the optimum path. This ideology will be modeled in the trailing mode search equation. The path labeled X_{inf} is an infeasible path, which leads the agent to an infeasible solution. The movement towards this region is avoided by appropriately selecting a suitable value for the SDA control parameters.

During the search process, the SDA identifies the region with high concentration of smell by performing the sniffing mode, representing the canine's first position in Fig. 3. The agent updates its position by performing the entire process of SDA and updating (iteratively) its position until the smell source is reached (optimum solution is obtained).

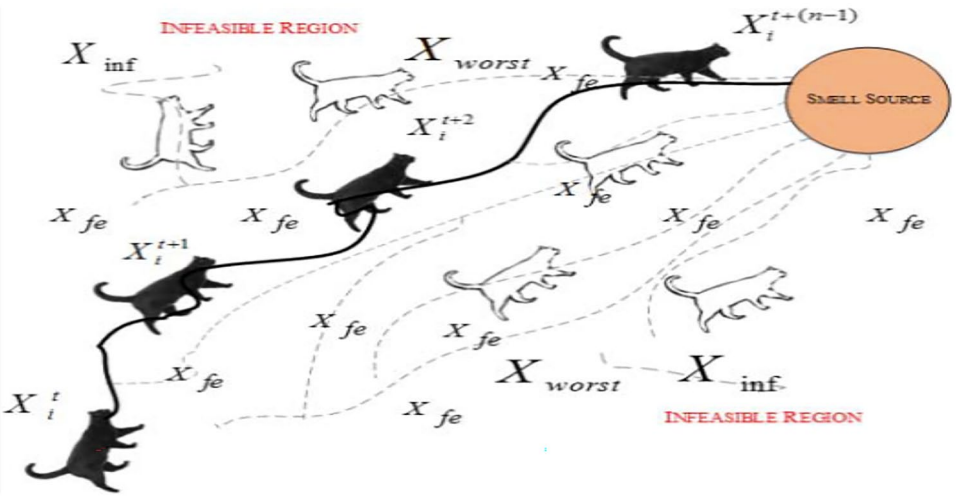

Fig. 3 Conceptual framework of smell detection algorithm

Random Mode The smell molecules are discrete if these molecules are separated by a large distance apart from the molecular search dimension. The intensity/ concentration of the smell molecule varied over time from one point to another. This bewilders the agent, and the agent may subsequently lose the smell, making the trail a challenge. At this point, the agent may be trapped into local minimal, leading to its inability to continue trailing. The agent moves randomly within the smell perception region in the natural situation, hoping to perceive the smell molecule again. This behaviour of the agent will be model as the random mode in the smell agent optimization.

3 Path Driven Testing

Path-driven testing examines the various routes that lead from the root to the destination node.At any rate once, all mixes of different choice or control articulations are executed. The strategy depends on the program's coherent design. During testing, a graph with all possible paths is drawn and confirmed.

Control Flow Graph (CFG) A CFG represents the logical complexity of the programme module to be assessed. There are various nodes and edges in the CFG. The edges represent the flow of control between the nodes, whereas the nodes represent executable code lines. A CFG diagram is used to generate all efficient paths.

Cyclomatic Complexity The most extreme number of potential ways in a diagram with M predicate hubs is 2M, and if the CFG has any circling articulations, there will be an innumerable number of test ways. The factor of cyclomatic intricacy number is a significant boundary to limit the complete check of possible test ways. Cyclomatic intricacy number is fundamental for the approval of directly free test ways in a chart. There are two elements related to a CFG: the cyclomatic number signified by 'V in chart hypothesis. The other is the intricacy esteem 'G' as an element of the diagram.

Analyzers mean to assess every one of the achievable ways in the CFG. The significant test in testing is to track down the ideal and plausible ways. Thus, to track down the ideal way, a need positioning is accomplished for every practical way. The most elevated need way will be first chosen for testing and proceeds until the least need way is tried.

The technique of premise way testing:

A product module contains different freeways to be tried. This load of ways ought to be tried in any event once in fundamental way testing.

Following are the different strides of testing measure.

1. Develop the CFG of the program module to be assessed.
2. Determine the cyclomatic intricacy of the CFG, for tracking down the conceivable number of straightly autonomous test ways.
3. Create sets of premise test ways utilizing the standard strategy:
   a Select the main practical autonomous way to be tried.
   b Back follow the autonomous ways by unexpectedly moving to each predicate hub to make fresher ways.

3.1 SDA Algorithm for Path Testing

The considered issue has a source and objective with numerous ways crossing. At the point when the quantity of ways builds, the intricacy of critical thinking additionally increments. The issue tackled by this calculation is addressed as a way found by various SDAs, beginning from a source navigating through the arrangement of smell spots and arriving at the objective. A smell spot is set apart by an SDA as its region when it is visited, and thus, the other SDAs can't visit the stamped spots. This guarantees that unmistakable ways are acquired from the SDAs. Different ways can be acquired by reproducing the SDA when it is hit with various similarly ideal ways. The replication of the SDA can be an answer for this. This would involve getting ways to cover and guarantee that any conceivable 'neighbourhood minima' stay away. The climate is instated by haphazardly chosen smell spots and appointing smell esteems conversely relative to the Cartesian distance between the source and the objective.

Two information structures are utilized for SDAs and smell spots. All specialists are introduced with signature esteems, and sweep esteems that are put at the beginning area. Each specialist is iteratively permitted to study the region inside its sweep to choose a plain smell spot with a high smell worth visiting. The SDAs are moved to picked spots, and the progression is rehashed until the SDAs arrive at the objective. It is expected that the SDAs have diverse smell location limits, which makes a viable hunt starting with one smell point then onto the next. Allow us to expect that a smell spot is 'stamped' when an SDA is visited to upgrade this calculation. The best way acquired by an SDA is registered. The fundamental strides of the proposed procedure are given in Algorithm 1.

Algorithm 1 : Smell Detection Algorithm

Input: Number of smell detection agent, N_1 with N_2 number of smell spots
N_3 is the number of smell agents that can reach the destination
Output: Agent Count

Let
N_1 : Number of SDAs
N_2 : Number of smell spots
N_3 : Number of SDAs that can reach the destination

1. Initialize the agents with natural numbers as signature indices and radii value in the inverse order, so that the SDA is considered in the iteration that have highest radius.
2. Randomly select N_2 points within the extremities of the Cartesian plot as smell spots. Assign the smell value 's' to each smell spot,

$$s = 1 / (a + b \times d)$$

where 'd' is the Cartesian distance between the smell spot, the destination and 'a' and 'b' are proportionality constants.
3. Initialize each agent to the source point
4. For every agent from 1 to N_1
   4.1 Choose the unmarked point (within the radius) from a set of smell spots that has a highest smell value.
   4.2 Move the SDA to the point by marking value of SDA's signature in smell spot's index.
5. Repeat step 4 until the destination is reached
6. Count the SDAs that reached the destination and assign to N_3

After N_3 SDAs have arrived at the objective, pick the best way from the acquired ways to answer the issue's answer condition. No two SDAs can pick a similar way. When an SDA visits a smell spot, it is marked as visited so that other SDAs need to visit a similar smell spot. This means that, if a node or region is already visited by an agent, no other agent can choose the same path. Figure 4 shows how the agent moves. Let the red arrow shows the path in which an Agent 1 moved. When the next agent comes to Node N1, it indicates that N1 is already visited by another agent, so it moves to node N6. From N6, if it tries to move to N2, again the path is restricted as it was also visited by Agent 1. Thus Agent 2, need to find a new path which is not in the territory of Agent 1. Thus finally Agent 2, finds its own path to the destination via establishing a new territory.

An information structure can register the expense, and a hint of every way joined inside the SDA calculation. This thought can be utilized in multi-source objective issues.

4 Results and Discussion

The number of linearly independent paths through a programme module is measured by creating a Control Flow Graph of the code, which measures cyclomatic complexity. Reduce the cyclomatic complexity of the programme to reduce the risk of modification and make it easier to understand. For better understanding, a software programme "test" that employs switch case structures is provided below. Corresponding flow graph is shown in Fig. 5

```
// Procedure Name : test
input(z)
if (z<0) then
        output("no result");
else
        output("obtained result")
endif
switch(x)
        case 1: print("SUN")
        break;
        case 2: print("MON")
        break;
        case 3: print("TUE")
        break;
        default: print("OTHER")
end switch;
```

Cyclomatic complexity factor, V (G) = Number of Edges - Number of Nodes + 2,
Thus, V (G) = 16 - 13 + 2 = 5.
Now, let us infer the Test Routes obtained

Test Route TR1 : Entry -> a -> b -> c -> J.1 -> e -> f -> J.2 -> Exit
Test Route TR2 : Entry -> a -> b -> c -> J.1 -> e -> g -> h -> J.2 -> Exit

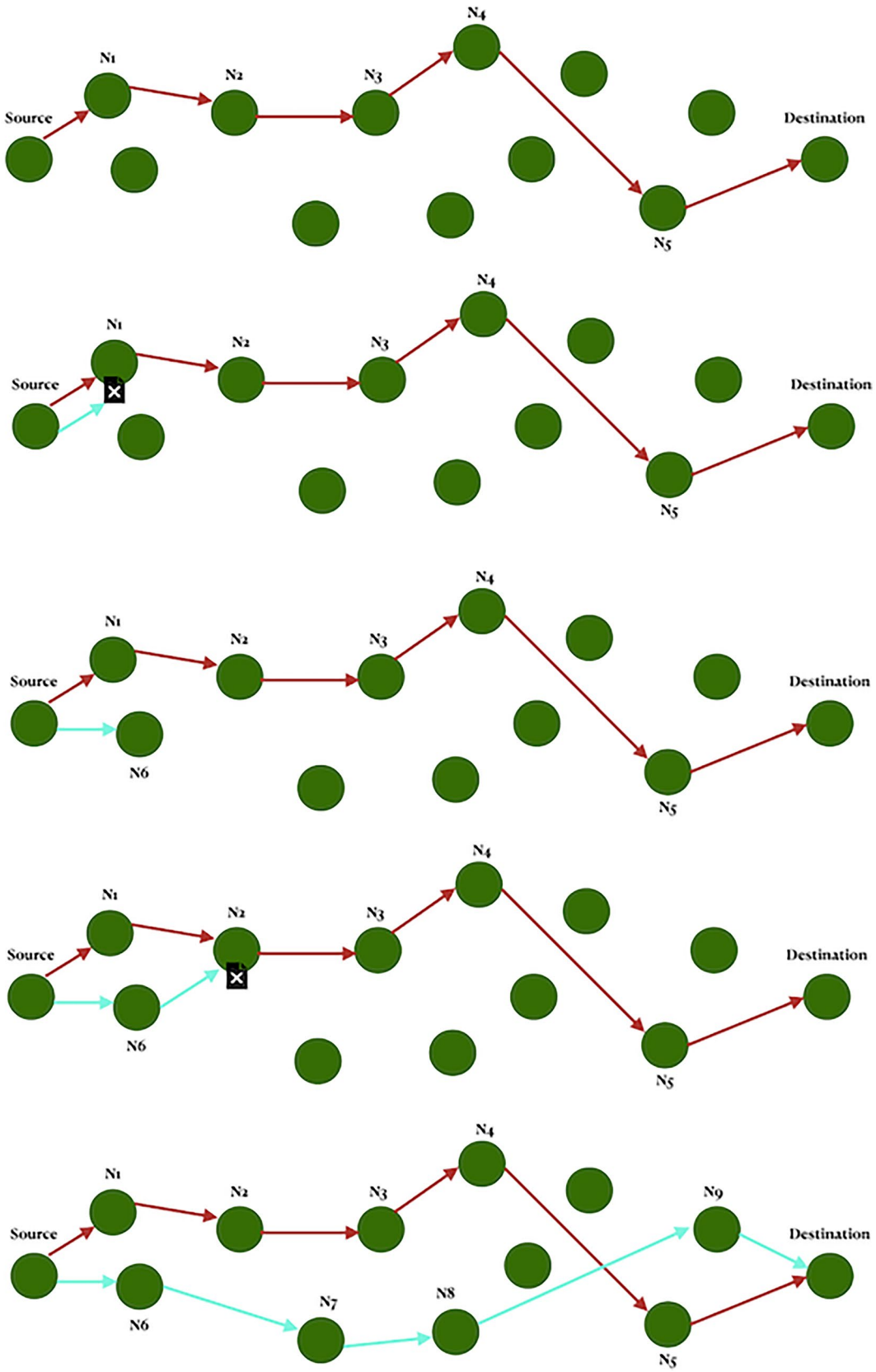

Fig. 4 Movement of agents

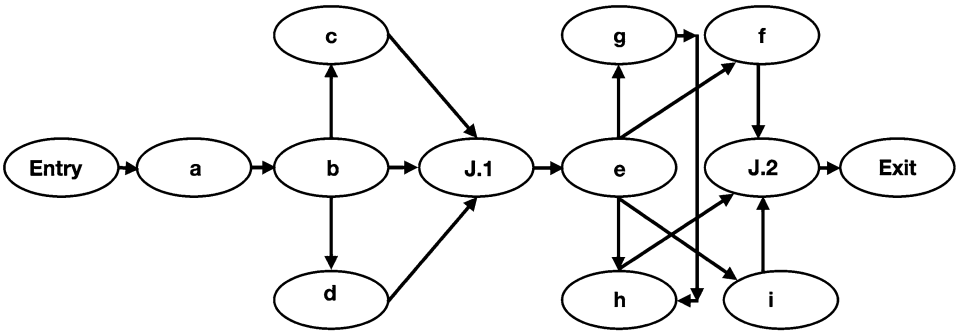

Fig. 5 CFG of program 'test'

Test Route TR3 : Entry -> a -> b -> c -> J.1 -> e -> h -> J.2 -> Exit
Test Route TR4 : Entry -> a -> b -> c -> J.1 -> e -> i -> J.2 -> Exit
Test Route TR5 : Entry ->a ->b -> d -> J.1 -> e -> f -> J.2 -> Exit
Test Route TR6 : Entry ->a -> b -> d -> J.1 -> e -> g -> h -> J.2 -> Exit
Test Route TR7 : Entry -> a -> b -> d -> J.1 -> e -> h -> J.2 -> Exit
Test Route TR8 : Entry -> a -> b -> d -> J.1 -> e -> i -> J.2 -> Exit

In this model, quantity of ways distinguished is eight (because of the utilization of switch case build). Since the cyclomatic intricacy got is just five, we need to get rid of infeasible ways. Utilizing the insect settlement advancement calculation gives the plausible ways and focused on the achievable ways. The calculation utilized the variables like way achievability, past experience, way perceivable and the visited status of the way. The model is included as a coordinated chart approach and the model additionally means the framework to be tried. This shows the different test ways of the model during its execution. The best arrangement of the way is made consequently after the execution of the streamlining calculation. The most noteworthy need way is chosen first and progressively, the wide range of various straight ways in the control stream diagram can be tried.

For creating the feasibly routes, there are mainly two steps.

i) generate a subset of feasible paths 'P' which hold true for the coverage criteria.
ii) from 'P', derive a subset 'p' in such a way that 'p' holds true for test coverage.

Also 2 criterion need to hold true by a route to belong in the set of basis path

a) the routes generated should be feasible in all manner so that there is an end-end connectivity and
b) the routes need to be linearly independent

Usecase: Now let's see a real-time code analysis using a test path. Consider the program 'test_pow', where we have three control structures that define the flow of execution.

```
// Program Name : test_pow
int a, b, pwr;
float c;
read(a, b);
if(y<0) then
        pwr = -b;
else
         pwr = b;
endif
c=1;
while(pwr!=0)
        c=c*a;
        pwr=pwr-1;
end while;
if(b<0) then
        c=1/c;
        write(c);
endif
```

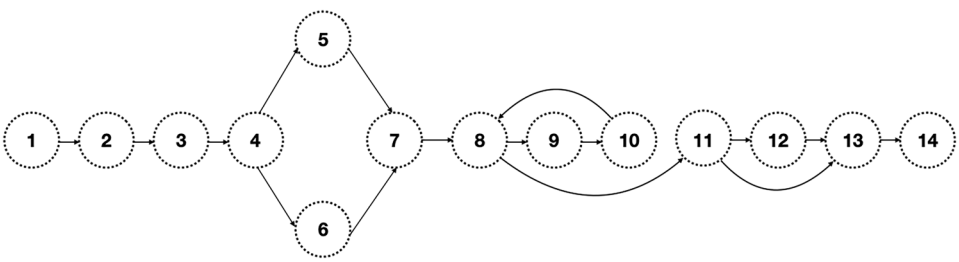

Fig. 6 Flow Diagram of program 'tes_pow'

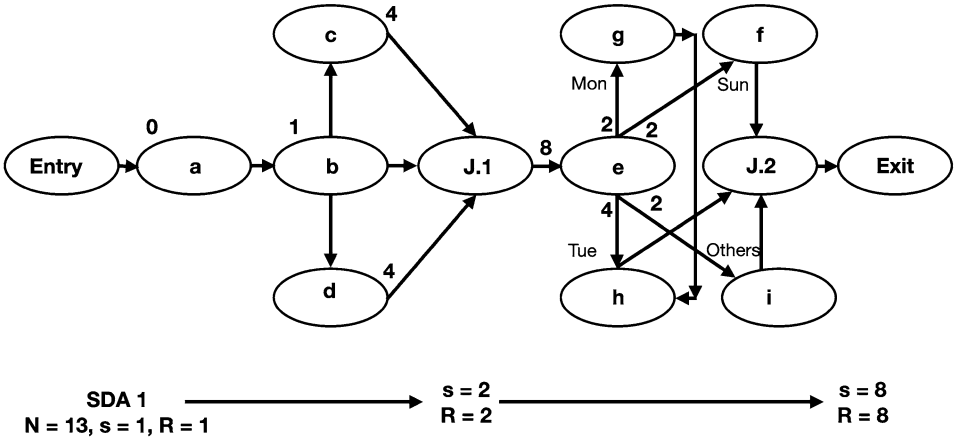

Fig. 7 CFG of program 'test' using SDA algorithm

A corresponding flow graph of the pseudocode can be depicted as in Fig. 6. Using the flow graph, the Cyclomatic Complexity(CC) may be calculated using the test path like,

CC factor, V (G) = Number of Edges - Number of Nodes + 2,

Test Route TR1 : Entry -> 1-> 2-> 3-> 4->6->7->8-> 11->12->13->14-> Exit
Test Route TR2 : Entry -> 1-> 2-> 3-> 4->5->7->8-> 11->12->13->14-> Exit
Test Route TR3 : Entry -> 1-> 2-> 3-> 4->6->7->8-> 11->13->14-> Exit
Test Route TR4 : Entry -> 1-> 2-> 3-> 4->5->7->8-> 11->13->14-> Exit

V (G) = 16 - 14 + 2 = 4.

4.1 Discussion

SDA discovers a path from an assortment set of smell spots from root to the objective hub. For 'n' specialists, there will be 'n' ways returned by the calculation. The doable ways are focused on from these 'n' ways. Additionally, the last number of hubs is likewise gotten. The underlying smell worth of every hub is contained in the hub area facilitates. The qualities get refreshed while navigating from the source to the objective. Recognizable proof of the following source and objective hubs will give the best way. To estimate the smell worth of every hub from the objective hub, the upsides of starting smell, decrement tally, which is converse of aggregate and compelling distance, are thought of. The upsides of smell are refreshed; all the SDA's are instated with ID esteem, current hub and length. Because of the smell worth of every hub, each SDA discovers a way.

The way is distinguished by considering the hub with the most noteworthy smell esteem from the current hub. This recognizable proof outcomes in relegating the most elevated smell hub as the current hub, and this circling cycle will proceed until the objective is reached. The SDA is relegated with a banner 'stop', when the SDA shows up at the objective. The extraordinary ways are distinguished from the SDA's who have shown up at the objective with the most elevated smell esteem. The streamlined way is found by

Table 1 Comparison Chart

Parameters	**Algorithm**	
	ACO	**SDA**
Tally of independent paths	All independent paths are priorited with equal threshold.	Identified paths are given priority in a ranked manner from 1 to 8.
Swarm relay	Using stigmergy ants communicate.	Territory based communication is done in canine.
Algorithm applicability	ACO is used to solve when the source and destination are already defined.	For obtaining optimum solution multiple agents are deployed.
Problem statement	Solution space need to be spotted, so a construction graph is used for doing the same.	Value specific Cartesian plot with accurate measure is used to spot the area.
Complexity Representation	$O(n^2)$	O(E + V log V)

Table 2 Time Complexity

Sl No	Application	Algorithm	
		ACO	SDA
1	Triangle Classification Problem	23.28	12.00
2	CVM	25.90	13.25
3	Examination	35.67	16.59
4	Job Portal	43.64	22.45
5	Online Application	50.28	25.69
6	MIS	57.86	27.98
7	Travel	61.23	30.05
8	Admission	40.27	19.86
9	Job Scheduling	70.00	40.00
10	Salary	78.00	49.95

contrasting the all outnumber of hubs visited by each SDA. For CFG, the number of hubs, weight allowed to every hub, greatest smell worth and most extreme span are thought of. Each edge's weight corresponds to the most extreme number of times; an SDA visits every hub. The need is top for the one of a kind way having the most extreme smell esteem and relies upon the weight allocated to each edge.

In the CFG of the model program "test", the SDA calculation fills in as follows. At first, hubs N = 13, beginning smell esteem, s =1, a tally of SDAs, and N1=1. The tally of smell spots denoted by N2 which is equal to 13 (Same as that of N2), sweep or distance from the source is at first zero. Figure 7 gives the proposed calculation for the model program "test". According to the SDA calculation, the essential ways are navigated by the SDA and fix the need. In premise way testing, all ways should be tried. However, one test engineer can't know about every one of the significant ways. The SDA calculation proposed in the model gives every recognised doable way in the request for need. Each edge of the CFG has a weight that relies upon the smell spot esteem. Additionally, SDAs distinguish all the test ways in a CFG. In the above program 'test', the need astute rundown of way testing is given beneath.

Test Route 1 : Entry -> a -> b -> d -> J.1 -> e -> h -> J.2 -> Exit
Test Route 2 : Entry -> a -> b -> d -> J.1 -> e -> g -> h -> J.2 ->Exit
Test Route 3 : Entry -> a -> b -> d -> J.1 -> e -> f -> J.2 -> Exit
Test Route 4 : Entry -> a -> b -> d -> J.1 -> e -> i -> J.2 -> Exit
Test Route 5 : Entry -> a -> b -> c -> J.1 -> e -> h -> J.2 -> Exit
Test Route 6 : Entry -> a -> b -> c -> J.1 -> e -> g -> h -> J.2 -> Exit
Test Route 7 : Entry -> a -> b -> c -> J.1 -> e -> f -> J.2 -> Exit
Test Route 8 : Entry -> a -> b -> c -> J.1 -> e -> i -> J.2 -> Exit

In calculation, the ways are chosen arbitrarily for age from source to the objective. In our SDA calculation, the choice of ideal ways depends on the greatest weight doled out to each most crossed edge of the CFG. In ACO calculation, the directing is done dependent on diminishing pheromone esteem. In SDA calculation, the olfactory capacity is expanding from source to the objective hub accordingly, the time intricacy is decreased.

A correlation of ACO and SDA calculations utilized in underlying testing is given underneath. Table 1 merges the outcomes in four distinct boundaries: the tally of autonomous ways, correspondence, materialness, and intricacy.

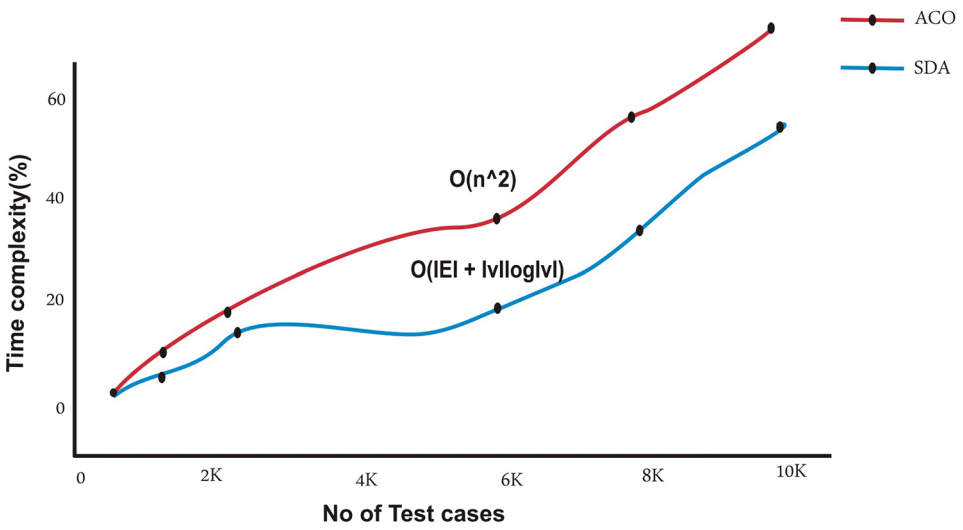

Fig. 8 Time Complexity Vs Test Cases

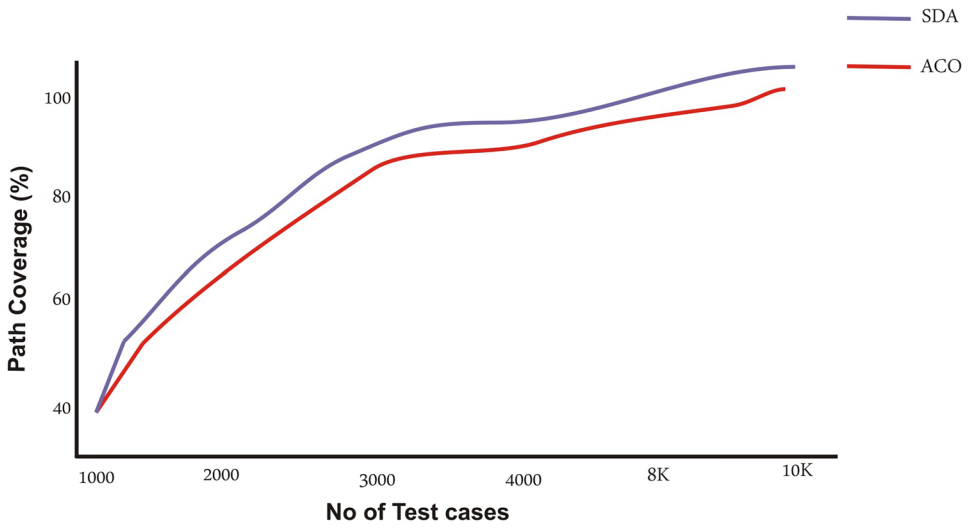

Fig. 9 Path coverage Vs Test Cases

Table 3 Path Coverage

Sl No	Application	Number of Testcases	Algorithm	
			ACO	SDA
1	Triangle Classification Problem	50	28.04	30.49
2	CVM	100	32.23	38.73
3	Examination	500	43.08	49.86
4	Job Portal	1500	67.06	72.65
5	Online Application	2000	81.63	83.87
6	MIS	3000	92.65	93.90
7	Travel	3500	94.76	96.99
8	Admission	1000	54.09	60.87
9	Job Scheduling	4000	97.00	98.98
10	Salary	4500	98.34	99.10

The proposed algorithm has been tested with various applications for routing and estimation. Tables 2 and 3 shows complexity comparison and the path coverage obtained. Corresponding growth curves are also shown in Figs. 8 and 9.

5 Conclusion

The work showed test way creation strategies for premise way testing in this paper. It's anything but a reasonable advancement system for achievable test way age in underlying testing by utilizing SDA streamlining calculation. After carrying out this strategy, the algorithmic interaction chooses the best test way succession dependent on its need. The most elevated need test way is chosen first for test execution, and in quite a while, all the following need freeways in the control stream diagram can be tried. The SDA Algorithm will, in general, be more gainful for better way inclusion in premise way testing. The model keeps away from copy ways dependent on the SDA signature and the visited status of the hubs. The outcomes show that the SDA calculation based underlying testing can be reached out to create the ideal and focused on the age of test ways for multipath programming modules. Result shows a peak improvement in the complexity analysis, from $O(n^2)$ to O(E+V log V) A robotized strategy can be seen as a future scope of the work. An automated test case generation for synchronous application will have a lot of future prospects. That can also be seen as a trending research area for scholars working in those fields.

Acknowledgments Authors would like to thank, Government of India for issuing Copyright for the algorithm Smell Detection Agent based Optimization Algorithm and Indian Patent to the work Bio-inspired Controller for Finding Disjoint Paths in Software Defined Networks, which were the basic concepts used in the development of the proposed work. The authors would also like to extend gratitude to all researchers affiliated with the Machine Intelligence Research(MIR) Laboratory for their support during each phase of this work.

Author Contributions Vinod developed the problem statement, conceptualized this study and developed the methodology. Anand has curated the data and was involved in the methodology implementation. Saju implemented and wrote the initial manuscript along with curation.

Funding Authors confirm that there is no funding received for this work.

Data Availability Supplementary materials are available in http://www.mirworks.in/downloads.php.

Declarations

Ethical Approval and Consent to Participate The work was done by following all ethical methods and authors declare the consent to participate in the work.

Consent for Publication All authors here by give consent for the publication of the article. There are no other person with competing interest.

Research Involving Human Participants and/or Animals The authors declare that this project does not involve Human Participants and/or animals in any capacity.

Informed Consent The authors declare that this research does not involve any surveys or participants in any capacity.

Competing Interests The authors have no competing interest to declare that are relevant to the work or content of this article.

Conflict of Interest Authors certify that this article has no actual or potential conflict of interest.

References

1. Abed-Alguni H. Bilal, Alawad Noor Aldeen (2021) Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments. Appl Soft Comput 102:107–113
2. Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-Qaness MA, Gandomi AH (2021) Aquila Optimizer: A novel meta-heuristic optimization algorithm. Comput Ind Eng 157:107–250
3. Abualigah Laith, Diabat Ali, Mirjalili Seyedali, Elaziz Mohamed Abd, Amir Gandomi H (2021) The Arithmetic Optimization Algorithm. Comput Methods Appl Mech Eng 76:113609
4. Abualigah Laith, Elaziz Mohamed Abd, Sumari Putra, Geem Zong Woo, Amir Gandomi H (2022) Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. Expert Syst Appl 191:116–158
5. Alawad NA, Abed-alguni BH (2021) Discrete Jaya with refraction learning and three mutation methods for the permutation flow shop scheduling problem. J Supercomputing 78:3517–3538
6. Alawad NA, Abed-alguni BH (2021) Discrete island-based cuckoo search with highly disruptive polynomial mutation and opposition-based learning strategy for scheduling of workflow applications in cloud environments. Arab J Sci Eng 46:3213–3233
7. Ananthalakshmi Ammal R, Sajimon PC, Vinod Chandra SS (2020) Canine algorithm for node disjoint paths. Lect Notes Comput Sci 12145
8. Colorni A, Dorigo M, Maniezzo V, Trubian M (1999) Ant System for Job-Shop Scheduling. Belg J Oper Res Stat Comput Sci 34(1):39–53
9. Doerner K, Gutjahr WJ (2003) Extracting test sequences from a markov software usage model by ACO. Lect Notes Comput Sci 2724: 2465–2476
10. Dorigo M, Maniezzo V, Colorni A (1996) Ant System: Optimization by a Colony of Cooperating Agents. IEEE Trans Syst Man Cybern - Part B Cybern 26(1):29–41
11. Laith Abualigah (2020) Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering. Studies in Computational Intelligence book series 816:105746
12. Li K, Zhang Z, Liu W (2009) Automatic test data generation based on ant colony optimization. Proc. of Fifth International Conference on Natural Computation 216-219
13. Parpinelli RS, Lopes HS, Freitas AA (2002) Data mining with an ant colony optimization algorithm. IEEE Trans Evol Comput 6(4):321–332
14. Reimann M, Ulrich H (2006) Comparing backhauling strategies in vehicle routing using Ant Colony Optimization. CEJOR 14(2):105
15. Saju Sankar S, Vinod Chandra SS (2020) A multi-agent ACO algorithm for effective vehicular traffic management system. Lect Notes Comput Sci 12145:640–647
16. Saju Sankar S, Vinod Chandra SS (2020) A Structural Testing Model using SDA Algorithm. Lect Notes Comput Sci 12145:405–412
17. Saritha R, Vinod Chandra SS (2016) An approach using Particle Swarm Optimization and Rational Kernel for variable length data sequence optimization. Lect Notes Comput Sci 9712:401–409
18. Saritha R, Vinod Chandra SS (2017) Multi dimensional honey bee foraging algorithm based on optimal energy consumption. Journal of The Institution of Engineers (India): Series B 98(5), 527-531
19. Saritha R, Vinod Chandra SS (2018) Multi modal foraging by honey bees toward optimizing profits at multiple colonies. IEEE Intell Syst 34(1):14–22
20. Sharma B, Girdhar I, Taneja M, Basia P, Vadla S, Srivastava PR (2011) Software coverage : A testing approach through ant colony optimization. Lect Notes Comput Sci 7076
21. Singh Y, Kaur A, Suri B (2010) Test case prioritization using ant colony optimization. ACM SIGSOFT Software Engineering Notes: 35(4)
22. Srivastava P (2012) Optimal test sequence generation: an approach using ant colony optimisation. Int J Comput Syst Eng 1:91–99
23. Vinod Chandra SS (2015) Smell Detection Agent Based Optimization Algorithm. J Inst Eng: Series B 97(3):431–436
24. Vinod Chandra SS, Anand HS (2022) Nature inspired meta heuristic algorithms for optimization problems. Computing 104:251–269
25. Vinod Chandra SS, Anand Hareendran S (2021) Phototropic algorithm for global optimisation problems. Appl Intell 51:5965–5977
26. Vinod Chandra SS, Anand HS, Saju Sankar S (2020) Optimal Reservoir Optimization Using Multiobjective Genetic Algorithm. Lect Notes Comput Sci 12145:445–454
27. Wegener J, Baresel A, Sthamer H (2001) Evolutionary test environment for automatic structural testing. J Inf Softw Technol 43:841–854
28. Yang S, Man T, Xu J (2014) Improved ant algorithms for software testing cases generation. Sci World J

S. S. Vinod Chandra is a Professor in the Department of Computer Science, University of Kerala. Since 1999, he has taught in various Engineering Colleges in Kerala. He has obtained his Ph.D. from University of Kerala and M.Tech. in Software Engineering from Cochin University of Science and Technology, India. He has discovered four microRNAs in the human cell and has six patents in the machine learning field. He authored nine books and a modest number of research publications. He is a reviewer of many international journals and conferences. His research area includes machine intelligence algorithms and nature-inspired computing. He is heading Machine Intelligence Research (MIR) Laboratory, a pinpoint research group in machine intelligence and nature-inspired techniques. He is leading many e-Governance projects associated with Universities and Government. He holds many consultancy activities for Government organisations.

S. Saju Sankar received his Ph.D. degree from University of Kerala and Master's degree in Software Engineering from Cochin University of Science and Technology, India. He is currently working as the Head of the Department at Government polytechnic college, Punalur. His prime research areas include optimisation, software automation, machine learning algorithms and computer vision.

H. S. Anand is currently working as Associate Professor at Muthoot Institute of Technology and Science, Kochi. He obtained his Ph.D. degree in Computer Science from University of Kerala and M.Tech. degree from

Anna University, Chennai. His current area of research includes machine learning algorithms, association rule mining and bioinspired methodologies. He has authored four books, many research journal publications, and patents in machine learning algorithms. He has designed and implemented various rule mining algorithms, which find application in the medical field, route mapping and frequent item search. He is a reviewer in various international conferences and journals. He is an active member of Machine Intelligence Research group.