Proposed Software Testing Using Intelligent techniques (Intelligent Water Drop (IWD) and Ant Colony Optimization Algorithm (ACO))

Laheeb M. Alzubaidy [1], **Baraa S. Alhafid** [2]

[1] **Software Engineering, Mosul University, Collage of Computer sc. And Mathematic Mosul , Iraq**

[2] **Software Engineering, Mosul University, Collage of Computer sc. And Mathematic Mosul , Iraq**

Abstract

This paper proposed software testing system by using artificial intelligent techniques. And that was conducted through Suggestion Intelligent Water Drop Algorithm (IWD) with white box testing for generated basis bath testing and using Ant Colony Optimization Algorithm (ACO) for test data generation. Correctly generated Test data helps in reducing the effort while testing the software. Automatic generation of test data is required to enables the corporation which develops the program to save time and costs as well as ensuring the test process quality, which is estimated by 50% of the product cost.

***Keywords**: Artificial Intelligent Techniques, Intelligent Water Drop (IWD), Ant Colony Optimization (ACO), Software Testing, Path Basis Testing, Test Data Generation.*

1. Introduction

Software test is the main approach to find errors and defects assuring the quality of software. Software testing is an expensive component of software development and maintenance. Testing is a complex, labor-intensive, and time consuming task that accounts for approximately 50% of the cost of a software system development [16]. Aim of the software testing is to uncover errors and faults present in the program, so that customer requirement can be properly fulfilled. Testing phase includes in the review of specification, analysis, design, and implementation part of the Software Development Life Cycle (SDLC). Manual generation of test data for testing the program, results in low reliability and high cost [12]. Due to the lack of cost and reliability, automation of testing process is necessary, so that the cost of testing can be reduced. Artificial Intelligence (AI) based techniques can help in removing this situation. AI based technique helps in solving the problem by using fast and proper judgments rather than using step by step deduction [2].
The paper is structured as follows: section 2 introduces related work, section 3 present testing in software engineering with the objective and type of software testing, section 4 describes the intelligent water drop algorithm, section 5 describes the ant colony optimization algorithm, section 6 includes the proposed work, section 7 includes Conclusions.

2. Related work

Various techniques have been proposed for automated testing to reduce efforts to a remarkable extent.

Andreas W., Stefan W., Joachim W.in (2007) suggested an empirical comparison of a genetic algorithm and a particle swarm algorithm applied to evolutionary structural testing. They selected 25 artificial test objects that cover a broad variety of search space characteristics (e.g. varying number of local optima), and 13 industrial test objects taken from various development project. The results indicate that particle swarm optimization is well-suited as a search engine for evolutionary structural testing and tends to outperform genetic algorithms in terms of code coverage achieved by the delivered test cases and the number of needed evaluations [1].

Praveen R. S., Tai-hoon K.in (2009) presents a method for optimizing software testing efficiency by identifying the most critical path clusters in a program. They do this by developing variable length Genetic Algorithms that optimize and select the software path clusters which are weighted in accordance with the criticality of the path. Exhaustive software testing is rarely possible because it becomes intractable for even medium sized software. Typically only parts of a program can be tested, but these parts are not necessarily the most error prone. Therefore, they are developing a more selective approach to testing by focusing on those parts that are most critical so that these paths can be tested first. By identifying the most critical paths, the testing efficiency can be increased [11].

Surender S. D., Jitender K. C. , Shakti K. in (2010) presents an artificial bee colony based novel search technique for automatic generation of structural software tests. Test cases are symbolically generated by measuring fitness of individuals with the help of branch distance based objective function. Evaluation of the test generator was performed using ten real world programs. Some of these programs had large ranges for input variables. Results show that the new technique is a reasonable alternative for test data generation, but doesn't perform very well for large inputs and where constraints are having many equality constraints [15].

Sanjay S., Dharminder K., H M Rai and Priti S. in (2011) presents a technique that based on a combination of genetic algorithm (GA) and particle swarm optimization (PSO), and is thus called GPSCA (Genetic-Particle Swarm Combined Algorithm) which is used to generate automatic test data for data flow coverage with using dominance concept between two nodes. The performance of the proposed approach is analyzed on a number of programs having different size and complexity. Finally, the performance of GPSCA is compared to both GA and PSO for generation of automatic test cases to demonstrate its superiority [16].

3. Testing in software engineering

There are a many definitions of software testing, but one can shortly define that as: "**A process of executing a program with the goal of finding errors**". So, testing means that one inspects behavior of a program on a finite set of test cases (a set of inputs, execution preconditions, and expected outcomes developed for a particular objective [9].

The objectives of software testing are: [13]

- A good test case is one that has a high probability of finding an as-yet undiscovered error.
- A successful test is one that uncovers an as-yet undiscovered error.
- Testing is a process of executing a program with the intent of finding an error.

The Testing type in software engineering is:

- Black box Testing
- White box Testing
- Gray box Testing

3.1 White box Testing

In this paper a White box testing is used, White box testing based on an analysis of internal working and structure of a piece of software. White box testing is the process of giving the input to the system and checking how the system processes that input to generate the required output as illustrated in Fig 1 .It is necessary for a tester to have the full knowledge of the source code. White box testing is applicable at integration, unit and system levels of the software testing process. In white box testing one can be sure that all parts through the test objects are properly executed [10].

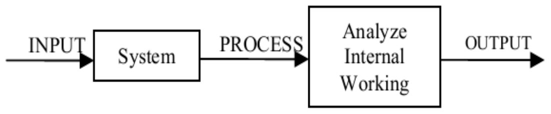

Fig. 1 Represent working process of White Box Testing

The types of white box testing techniques are : [10]

- Control Flow Testing
- Branch Testing
- Basis Path Testing
- Data Flow Testing
- Loop Testing

3.1.1 Basis Path Testing

Basis path testing is a white-box testing technique first proposed by Tom McCabe [13] and it allows the test case designer to produce a logical complexity measure of procedural design and use this measure as an approach for outlining a basic set of execution path (basic set is the set of all the execution of a procedure) These are test cases that exercise basic set will execute every statement at least once. Basic path testing makes sure that each independent path through the code is taken in a predetermined order. For this reason Basis Path Testing is used in this paper.
The method devised by McCabe to carry out basis path testing has four Steps. These are [5]:

- Compute the program graph.
- Calculate the cyclomatic complexity.
- Select a basis set of paths.
- Generate test cases for each of these paths

a. Flow Graph Notation

Before we consider the basis path method, a simple notation for the representation of control flow called allow graph (or program graph) must be introduced, The flow graph depicts logical control flow using the notation illustrated in Fig 2, Each structured construct has a corresponding flow graph symbol [13].

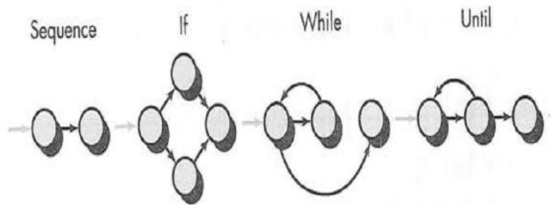

Fig 2 Flow Graph Notation.

Control Flow Graph (CFG) describes the sequence in which the statements/instructions of a program are executed. It is representation of flow of control through the program. CFG is directed graph in which each node is a program statement/basic block and each edge represents the flow of control between statement/basic blocks. A basic block is a sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without halt or possibly of branching except at the end [8]. In a CFG, a node including condition is called a predicate node as shown in Fig 3, and edges from the predicate node must converge at a certain node. Area defined by edges and nodes is referred to as region [13].

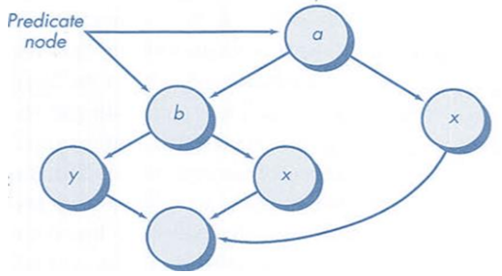

Fig 3 Predicate node.

On a flow Graph as shown in Fig 4 :

- the symbol arrows called as Edges that represent the flow of control
- Circles are called as nodes, which represent one or more actions.
- Areas bounded by edges and nodes called regions

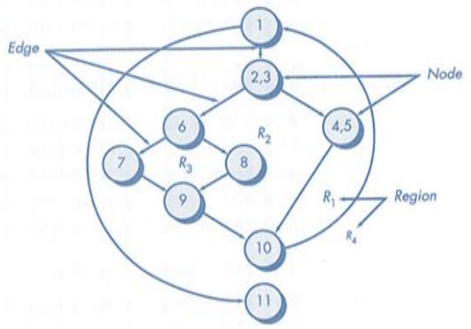

Fig4 Flow Graph .

b. Cyclomatic Complexity (CC)

The notion of Cyclomatic complexity was presented by McCabe. Cyclomatic complexity is software metric that delivers a quantitative degree of the logical difficulty of a program. Cyclomatic Complexity (CYC) is derived as the number of edges of the program's control-flow graph minus the number of its nodes plus two times the number of its linked components. Cyclomatic complexity purely depends on the Control Flow Graph (CFG) of the program to be tested [14] complexity is computed in one of three ways [13]:

- The number of regions of the flow graph corresponds to the Cyclomatlc complexity.
- Cyclomatic complexity V(G) for a flow graph G is defined as V(G)=E-N+2

Where E is the number of flow graph edges and N is the number of flow graph nodes.

- Cyclomatlc complexity V(G) for a flow graph G is also defined as v(G)=P+1

Where P is the number of predicate nodes contained in the flow graph G.

c. Determine Independent Paths

The value of V(G) Provides the upper bound on the number of linearly independent paths through the program Control structural . Through the Control flow graph in Fig 5 we expect to specify six Paths:
Path 1: 1-2-10-11-13
Path 2: l-2-10-12-13
Path 3: 1-2-3-10-11-13
Path 4: 1-2-3-4-5-8-9-2
Path 5: 1-2-3-4-5-6-S-9-2
Path 6: 1-2-3-4-5-6-7-8-9-2

It is often worthwhile to Identify predicate nodes as an aid in the derivation of test cases. In this case, nodes 2,3,5,6, and 10 are predicate nodes [13].

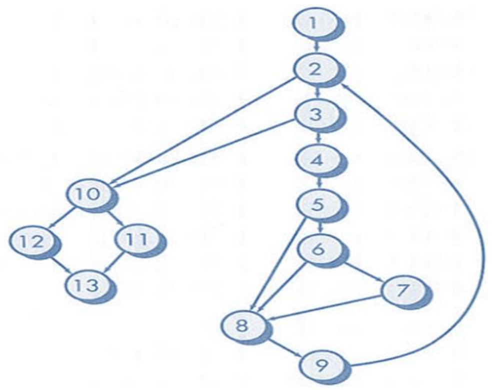

Fig 5 Control Flow Graph.

d. Deriving Test Cases

Data should be chosen so that conditions at the predicate nodes are appropriately set as each path is tested. Each test case is executed and compared to expected results. Once all test cases have been completed, the tester can be sure that all statements in the program have been executed at least once [13].

4. Intelligent Water Drop Algorithm

IWD algorithm [6][7] is a swarm-based optimization algorithm, simulated from observing natural water drops in river. IWD has been applied to various problems like Travelling Salesman Problem (TSP), N-queen puzzle, Multidimensional Knapsack Problem (MKP) , etc. These results have proved the significance of IWD algorithm over other swarm optimization algorithms. Another solution for TSP using IWD algorithm [7] is introduced where proposed algorithm converges very fast to the optimum solution.The improved IWD algorithm [4] has been applied to solve the air robot path planning in dynamic environments and results are quite impressive over genetic algorithm and ACO algorithm Since IWD has not yet been applied to the area of software testing and the effective results have been produced for various problems, this paper tries to derive a solution model for software testing using IWD in the hope that expected results will be more significant than the current solutions available for test data generation. Before moving to the proposed solution of IWD, general introduction is provided which describes its strategy along with available metrics in it.

IWD algorithm is a new swarm-based optimization algorithm inspired from natural rivers. In a natural river, water drops move towards center of the earth, due to some gravitational force acting on it. Due to this the water drop follows the straight and the shortest path to its destination [6].Pictorial representation of basic IWD is shown in Fig 6. In ideal conditions it is observed that the optimal path will be obtained. Water drop flowing in the river has some velocity which is affected by another actor, i.e., soil.

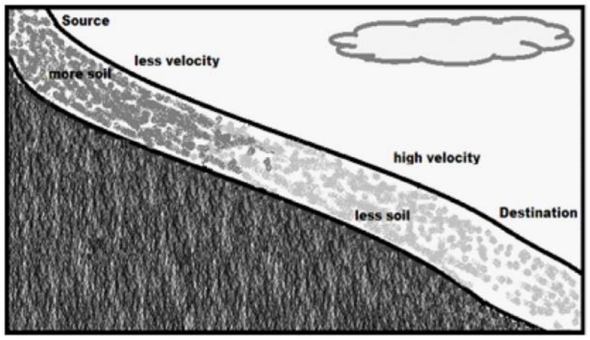

Fig 6 Pictorial representation of IWD.

Some changes that occurred while transition of water drop from one point to another point are:

1. Velocity of water drop is increased.
2. Soil content in the water drop is also increased.
3. Amount of soil in the riverbed from source to destination get decreased.

Water drop in the river picks up some soil in it when its velocity gets high and it releases the soil content when its velocity is less [7] Some of the prominent properties of the natural water drop are taken, based on which IWD is suggested. IWD has the two following important properties,

1. The amount of soil the water drop carries, which is represented by Soil (IWD) (or soil ^{IWD}).
2. The velocity of water drop with which it is moving now, denoted by Velocity (IWD) (or vel ^{IWD})

Value of both the properties may change during the transition. Environment contains lots of paths from source to destination [4] which may be known or unknown .When the destination is known, IWD follows the best path to reach the destination (best is in terms of cost and any other desired measure .(When destination is unknown it finds the

optimal destination. From the current location to the next location Velocity (IWD) is increased by an amount ‹which is nonlinearly proportional to the inverse of the amount of soil between the two locations ‹referred to as the change in velocity. The Soil (IWD), is also increased by extracting some soil of the path between two locations. The amount of soil added to the IWD is inversely (and nonlinearly) proportional to the time needed for the IWD to pass from its current location to next location. IWD chooses the path with less soil content. In the proposed approach, IWD is applied over the Control Flow Graph (CFG) to obtain the number of paths available in the program .The CFG depicts the logical control flow of the program [13]. All linearly independent paths could be obtained by CFG .Independent path is the path in the program that determines at least one new set of processing statement. In other words it introduces at least one new edge in the graph. Number of available paths can be obtained by finding the Cyclomatic complexity of the graph [13].
The IWD algorithm as specified by Shah-Hosseini H. in [6] is as follows:

1. Initialization of static parameters.
2. Initialization of dynamic parameters.
3. Spread the IWDs randomly on the nodes of the graph.
4. Update the visited node list of each IWD.
5. Repeat Steps a to d for those IWDs with partial solutions.
   a. For the IWD residing in node i, choose the next node j, which does not violate any constraints of the problem and is not in the visited node list of the IWD.
   b. For each IWD moving from node i to node j, Update its velocity.
   c. Compute the soil.
   d. Update the soil.
6. Find the iteration-best solution from all the solutions found by the IWDs.
7. Update the soils on the paths that form the current iteration best solution .
8. Update the total best solution by the current iteration best solution.
9. Increment the iteration number
10. Stops with the total best solution.

5. Ant colony Optimization Algorithm

The inspiring source of ACO is the food foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During their return trip, ants deposit a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. The main principle behind these interactions is called stigmergy , or communication through the environment. An example is pheromone laying on trails followed by ants [3].
Pheromone is a potent form of hormone that can be sensed by ants while traveling along trails. It attracts ants and therefore ants tend to follow trails that have high pheromone concentrations. This causes an autocatalytic reaction, i.e., one that is accelerated by itself . Ants attracted by the pheromone will lie more of the same on the same trail, causing even more ants to be attracted see Fig7. This characteristic makes swarm intelligence very attractive for network routing, robotics, optimization etc. A number of extensions are proposed to the original ant algorithm. These algorithms performed better producing much improved results than the original ant algorithm [3].

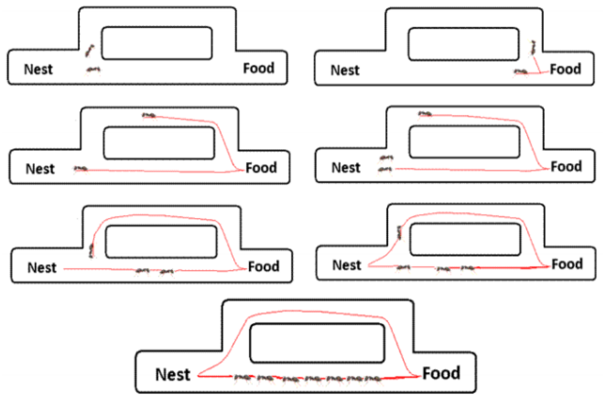

Fig 7 Optimization by Ant Colony.

Main characteristics of this model are positive feedback, distributed computation, and the use of a constructive greedy heuristic. The basic algorithm introduced by Marco Dorigo is given by following steps [3]:

1. Set parameters, initialize the pheromone trails
2. while (termination condition not met) do

a. Construct ants solutions
b. Apply local search
c. Update pheromones
d. end while

6. Proposed work

In the proposed work, it will be create a parser that used to convert input program to the corresponding control flow graph (CFG) for the program, then after knowing the benefits of intelligent water drop algorithm and their ability to find optimal solutions efficiently, it will be proposed to use it in the field of software testing through use in the generation of independent paths of the program and then, use ant colony optimization algorithm to generate the best test data that will be used in order to test all the independent paths the program and make sure it's have passed them all, as shown in Fig 8**.**

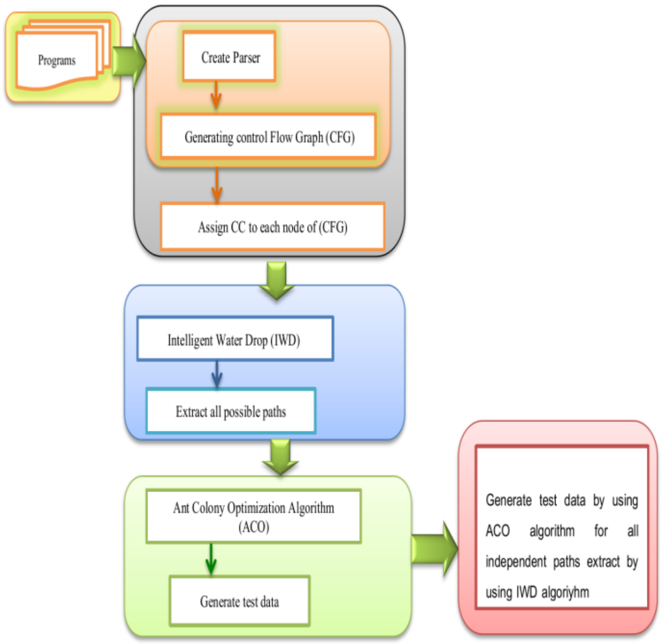
Programs
Create Parser
Generating control Flow Graph (CFG)
Assign CC to each node of (CFG)
Intelligent Water Drop (IWD)
Extract all possible paths
Ant Colony Optimization Algorithm (ACO)
Generate test data
Generate test data by using ACO algorithm for all independent paths extract by using IWD algoriyhm

Fig. 8 proposed work

7. Conclusions

After a thorough study of swarm intelligence and its branches in particular IWD algorithm, know the benefits of IWD in the field of computer science and superiority over other techniques in this field, where it proved its ability by testing many of the real problems. So we proposed a method in the software testing process as they avoided the consumption of a large number of duplicates and their ability to reach solutions ideal and efficient manner as well as having knowledge of the importance of ACO algorithm and its ability to generate data that are used in software testing process, for all these reasons it will design a system that uses an Intelligent Water Drop Algorithm (IWD) to generate independent paths and test data in order to test all Independent paths in the program and make sure it's have passed them and covered efficiently.

References

[1] A. Windisch, S. Wappler, J. Wegener, "Applying Particle Swarm Optimization to Software Testing", ACM, 2007, pp. 1121-1128.
[2] F. N. Raza, , "Artificial Intelligence Techniques in Software Engineering"", In Proceedings of the International Multi Conference of Engineers and Computer Scientists, Vol.2174,2009, pp.1086-1088 .
[3] H. Afaq1 , S. Saini , " On the Solutions to the Travelling Salesman Problem using Nature Inspired Computing Techniques", IJCSI International Journal of Computer Science Issues, Vol. 8, No. 2,2011, pp. 326-334.
[4] H. Duan, S. Liu, X. Lei," Air Robot Path Planning Based on Intelligent Water Drops Optimization". IEEE International Joint Conference on Neural Networks , 2008, pp. 1397-1401.
[5] H. Schligloff , M. Roggenbach, " Path Testing" , Advanced Topics in Computer Science: Testing, citeseer ,2002.
[6] H. Sh. Hosseini ," The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm", IJBIC International Journal of Bio-Inspired Computation, Vol. 1, No. 1/2, 2009,pp. 71-79.
[7] H. Sh. Hosseini, "Problem solving by intelligent water drops". IEEE Congress on Evolutionary Computation, 2007, pp. 3226-3231.
[8] H. Tahbildar ,B. Kalita , "AUTOMATED SOFTWARE TEST DATA GENERATION: DIRECTION OF RESEARCH", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.1, 2011,pp. 99-120.
[9] Jovanovic , Irena," Software Testing Methods and Techniques",2008,pp.30-41.
[10] M. E. Khan, " Different Forms of Software Testing Techniques for Finding Errors", IJCSI International Journal of Computer Science Issues, Vol. 7, No 1, 2010, pp. 11-16.
[11] P. R. Srivastava , T. Kim," Application of Genetic Algorithm in Software Testing" ,International Journal of Software Engineering and Its Applications,Vol.3, No.4,2009,pp.87-96.
[12] R. S. Pressman , "Software Engineering A Practitioner's Approach, FIFTH EDITION",5th, McGraw-Hill Company,2001.
[13] R. S. Pressman , "Software Engineering A Practitioner's Approach, Seventh Edition",7th, McGraw-Hill Company ,2010.

[14] S. Nidhra, J. Dondeti," BLACK BOX AND WHITE BOX TESTING TECHNIQUES–A LITERATURE REVIEW" ,International Journal of Embedded Systems and Applications (IJESA) Vol.2, No.2, 2012,pp.29-50.

[15] S. S. Dahiya, J. K. Chhabra, Sh. Kumar , " Application of Artificial Bee Colony Algorithm to Software Testin21st Australian Software Engineering Conference, IEEE,2010,pp.149-154.

[16] S. Singla, D. Kumar, H M Rai, 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, 2011,pp.15-26

First Author :**Dr. laheeb M. Alzubaidy**, have BSc. In 1987, MSc. In 1992 And PhD in 2002, in computer Sc. From Dept. of computer Sc, university of Mosul, Iraq. Associative professor in 2003, Head of Dept of Computer Sc. In 2003, visiting lecturer in Isra private university in 2004, head of Dept of Software Engineerinh in 2007, Visiting lecturer in USM university , NAV6 center in 2009, interested research fields are in Artificial Intelligent technique, network security , image processing , pattern recognition, software eng.

Second Author : Baraa S. Alhafid , have BSc. In 2006, in Software Eng.. From Dept. of Software Eng., university of Mosul, Iraq. Researcher in 2011, MSc. Student in 2011 Dept. of Software Eng., university of Mosul, interested research fields are in Artificial Intelligent technique, software eng.