Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.029703
Article

Tech Science Press

A Hybridized Artificial Neural Network for Automated Software Test Oracle

K. Kamaraj[1,*], B. Lanitha[2], S. Karthic[3], P. N. Senthil Prakash[4] and R. Mahaveerakannan[5]

[1]KPR Institute of Engineering and Technology, Coimbatore, 641048, India
[2]Karpagam Academy of Higher Education, Coimbatore, 641021, India
[3]Hindusthan College of Engineering and Technology, Coimbatore, 641032, India
[4]RMK College of Engineering and Technology, Chennai, Tamil Nadu, India
[5]Saveetha School of Engineering, Saveetha University, Chennai, 602105, India
*Corresponding Author: K. Kamaraj. Email: Kamaraj.rj@gmail.com
Received: 10 March 2022; Accepted: 04 May 2022

Abstract: Software testing is the methodology of analyzing the nature of software to test if it works as anticipated so as to boost its reliability and quality. These two characteristics are very critical in the software applications of present times. When testers want to perform scenario evaluations, test oracles are generally employed in the third phase. Upon test case execution and test outcome generation, it is essential to validate the results so as to establish the software behavior's correctness. By choosing a feasible technique for the test case optimization and prioritization as along with an appropriate assessment of the application, leads to a reduction in the fault detection work with minimal loss of information and would also greatly reduce the cost for clearing up. A hybrid Particle Swarm Optimization (PSO) with Stochastic Diffusion Search (PSO-SDS) based Neural Network, and a hybrid Harmony Search with Stochastic Diffusion Search (HS-SDS) based Neural Network has been proposed in this work. Further to evaluate the performance, it is compared with PSO- SDS based artificial Neural Network (PSO-SDS ANN) and Artificial Neural Network (ANN). The Misclassification of correction output (MCO) of HS-SDS Neural Network is 6.37 for 5 iterations and is well suited for automated testing.

Keywords: Test oracles; neural network; particle swarm optimization; stochastic diffusion search; harmony search

1 Introduction

During the development of software, it passes several phases and software testing is a very crucial stage. Software testing is an assessment procedure carried out on a software product's reliability and quality. Faults in the software are identified through the use of software testing. Fault detection is done by numerous software testing methods which are utilized appropriately together with their automated tools. The success of the outcomes of the testing procedures largely relies on the manner in which the test procedure was performed. Software failure due to human error is a variation from the predicted functions of the software [1]. Compared to automated testing methods, testing the software manually requires a lot of manpower

assignments, which can take a lot of time and is also costly. Hence, software testers are extremely preferred due to higher reliability, and the testing procedure can be made quicker with less cost.

The software test oracle is a strategy to identify the incorrect and correct actions of particular Software Under Test (SUT) [2]. It is also described as a procedure for decision and strategy for the test results' interpretation. Upon execution of a certain test case on a SUT, the SUT outputs must be collected, and the outputs must also be identified as correct or incorrect in order to establish the software behavior's correctness. A comparison between the SUT's expected outputs and existing outputs can be used to distinguish the incorrect behavior and the correct behavior. The principal specification associated with any testing activity is the decision about an execution's correctness and revelation of the failures. Test oracles are very arduous work within the software test environment as the identification of a software's set of expected outputs it is chiefly reliant on SUT knowledge. In traditional methods, there is an "oracle problem" in certain cases as it is very arduous or not possible for determining the correctness of exits of tests or cases of tests. A human tester will analyze or assess the outputs or behaviors of the SUT. To obtain constructive software testing on a massive scale, testers can accomplish manual work reduction through automation of their test oracles [3].

In 1995, Kennedy et al. [4] put forward the PSO, an evolutionary method for computation that is influenced by the particle's social behavior. Every particle is allocated a velocity and a position, and this particle will traverse a multi-dimensional search space. Depending on the optimal position and the optimal particle location, the velocity will be adjusted at each iteration of the overall population. To balance the exploitation and exploration, the PSO will combine global search techniques with the local search techniques. PSO's are quite simple and can be used in an extensive application range with low cost for computation. Due to this, PSO has become quite well-known.

Stochastic Diffusion Search (SDS) is a novel probabilistic technique to recognize and match the best-fit. The SDS is a mode of distributed computation which utilizes the agents' simple interaction. SDS with mathematical foundation that is used to describe the algorithm's behavior by linear time complexity, convergence criteria to its minimum, stoutness, convergence at global optimum level, and resource assignment.

Another meta-heuristic global optimization algorithm is the harmony search algorithm [5]. The inspiration for this algorithm comes from the practice of music improvisation, by which a more harmonized state is sought by the musicians. For numerous optimization problems, the harmony search algorithm has been well utilized because of its flexibility, efficacy, and simplicity.

The novelty of the proposed work is the hybridization of the PSO and SDS to form an ANN, where the hybrid PSO-SDS optimizes the weights and biases of the ANN. The SDS is initialized with a set of randomly generated weights and biases assigned to the agents. Each agent is assigned to a combination of weights and biases (*i.e.*, hypothesis) from the search space. The search space represents all possible combinations of weights and biases.

The rest of this work is arranged into the subsequent chapters. The associated available works are discussed in Section 2. The various techniques proposed in this work are explained in chapter 3. The performance of the results of proposed approach are described in chapter 4, and to conclude, this work is brought to a close in chapter 5.

2 Related Work

For any software application, its efficient functionality analysis is the most significant event which decides the generated outcome's quality. Usually, there are high-quality assurance costs and high time consumption associated with analyzing scenarios, which consist of many test cases along with various

components. Various possible methods with their challenges and opportunities for testing process is analyzed in [6–8]. Kumar et al. [9] optimized test cases and prioritized it with the Particle Swarm Optimization algorithm (PSO) and then, the Improved Cuckoo Search algorithm (ICSA) was used. The result was later assessed for software quality actions.

Machine Learning (ML) based technique was proposed by Braga et al. [10] to automate the software's test oracle procedure. In this technique, an application's historical usage data is seized into the application being tested through the injection of a capture component. During the Database step this data traverse Knowledge is discovered. Later, these data are employed for training in order to produce an oracle that is feasible for the application under test. The assessment of the proposed technique was done using web applications to conduct four different experiments. The assessments demonstrated evidence of the technique's feasibility for the problem's solution.

Multi-Networks Oracles based on ANN are proposed by Shahamiri et al. [11] in order to automatically manage the mapping. It is actually an improved model of the earlier ANN-Based Oracles. Assessment on the specified scheme was carried out by a scheme given by code mutation and employed for the testing of two different case studies, which were industry-sized. A comparison of the proposed oracle's results was made with the earlier ANN-based Oracles. The proposed oracle had an accuracy of 98.93%, oracle was able to detect a maximum of 98% in case of injected faults. Compared to the earlier model, the evaluated metrics proved that the proposed oracle had better applicability and quality.

Testers can perform the testing procedure and fault detection using the guidance of test oracles, which are reliable and simple sources of anticipated software behavior. Study shows the comparison of two existing test oracles, IFN based regression tester, and Multi-network oracles based on ANNs, utilizing a black-box method, was conducted by Yousif et al. [12]. The study was used for the comparison of experimental studies, procedures, and assessment methodologies. Based on these outcomes, it was found that, compared to the IFN regression tester, the Multi-Network oracles have a low misclassification error rate of 1.74% and a superior accuracy rate of 98.26%. Thus, it has been found that for a software testing procedure, Multi-network oracles based on ANNs are more feasible and provide more reliability and quality. Another ANN based approach was contributed by Shahamiri et al. [13] which proved to be efficient in terms of accuracy and detection rate.

To improve the efficiency, Du et al. [14] modelled a method of generation of test case with multiple-path coverage, which is a combination of the PSO algorithm along with Metamorphic Relations (MRs). This scheme initially performs a new test case by making use of the MRs among test cases. The method further reduces all its evolving numbers in the PSO. The experimental result prove that the method proposed is able to enhance efficiency significantly in terms of evaluation of fitness and consumption of time. Sofokleous et al. [15] designed a dynamic framework involving a program analyzer to evaluate automatically. They used two schemes for test case generation. Even though the hit ratio is high this method is applicable only to Java environment. Xu et al. [16] constructed a decision tree from java byte code by transforming into Jimple format involving the predicates to make the decision. This model able to detect the injected fault upto 94.67%.

Test Case Prioritization (TCP) is a class of the NP-hard problem having a optimal solution that makes use of a soft computing approach. Ref. [17] had proposed another novel GA approach that solves the TCP issue. This algorithm is compared experimentally to the random technique. From this experiment, it had been identified that the Average Percentage of Fault Detection (APFD) for the GA provides better results compared to the random technique. Analysis had confirmed that the result was independent on the actual quantity of generations. A decision tree based scheme and ANN was designed in [18] for automated test oracle.

As there are challenges faced by the Specification-Based Testing (SBT), Yang [19] had proposed another new method for the efficient generation of test cases combining the GA with its formal specifications. The main work of this method was to reform the specifications using the GA for generating inputs to kill mutants of a prey code that is under assessment. For this scenario, there are two examples that were offered for competence demonstration. Results proved that the method could competently generate certain useful test cases for uncovering the mutant program contributing to the software maintenance. Hooda et al. [20] had made a proposal for another new concept proposed for making use of the UML state chart diagram along with test case generation tables. The ANN is a tool of optimization used to reduce redundancy for test case generation with the GA. There was a new strategy of minimization that is crucial for this, for which a Combinatorial Interaction Testing (CIT) test suite was used. Sangwan et al. [21] deigned a neural network involving radial basis method to generate test cases.

Altmemi et al. [22] included the validation and implementation of a strategy for t-way testing. The primary contribution of the Sine Cosine Algorithm SCA was to show the competitive nature of the strategy. Its primary contribution was to test data generation for high coverage strength ($t < 12$). Liu et al. [23] generated automated test cases using systematic functional scenario-based V model that uses divide and conquer strategy. This methodology is capable of imparting high path coverage. Zhang at al. [24] proposed probability based Neural network to perform automated testing and found to better in terms of speed and accuracy. The authors in [25] have used a hybrid approach of ANN with SDS to determine the weights of neural network there by improving classification accuracy. Even though there are multiple approaches being designed, these schemes still lacks in detecting the injected faults.

3 Methodology

3.1 Overview

In this section, the Stochastic Diffusion Search (SDS), Particle Swarm Optimization (PSO), Harmony Search (HS) Algorithm, Artificial Neural Network (ANN), Proposed PSO-SDS Algorithm and Proposed Harmony Search with SDS algorithm are detailed. Fig. 1 shows the proposed methodology of study carried out. The training is carried out with the 80% of the data and the testing is conducted with the 20% of the data. A 10 fold validation is used to validate the data.

3.2 Optimization of Stochastic Diffusion Search (SDS)

The population is initialized to begin the optimization or search in the SDS algorithm. Each agent will have a hypothesis h to define a potential problem solution in any SDS search. A hill is defined as the agent hypothesis in the analogy of game mining. The initialization phases are followed by the below two phases namely test and diffusion phase.

The SDS [26] will verify the success of the agent hypothesis results in a boolean value in the Test phase by carrying out a partial hypothesis evaluation. In the subsequent iteration, depending on the application of a precise recruitment strategy, the population can be diffused with successful hypotheses. Thus, this method can be used to convey information on possibly good solutions across a whole envoy population. Each envoy will carry out a partial function evaluation denoted by pFE in the Test phase. This pFE is a specific function of the hypothesis of the envoy given as $pFE = f(h)$. When applied in the mining game, rather than mining all the areas on the hill, partial function evaluation is used to mine an arbitrarily chosen area on the mountain. The hypothesis of the envoy will determine the arbitrarily chosen region. Each agent will employ another agent to interact with and to potentially communicate the hypothesis in the Diffusion phase. Diffusion is done through the communication of the hill hypothesis in the mining game analogy.

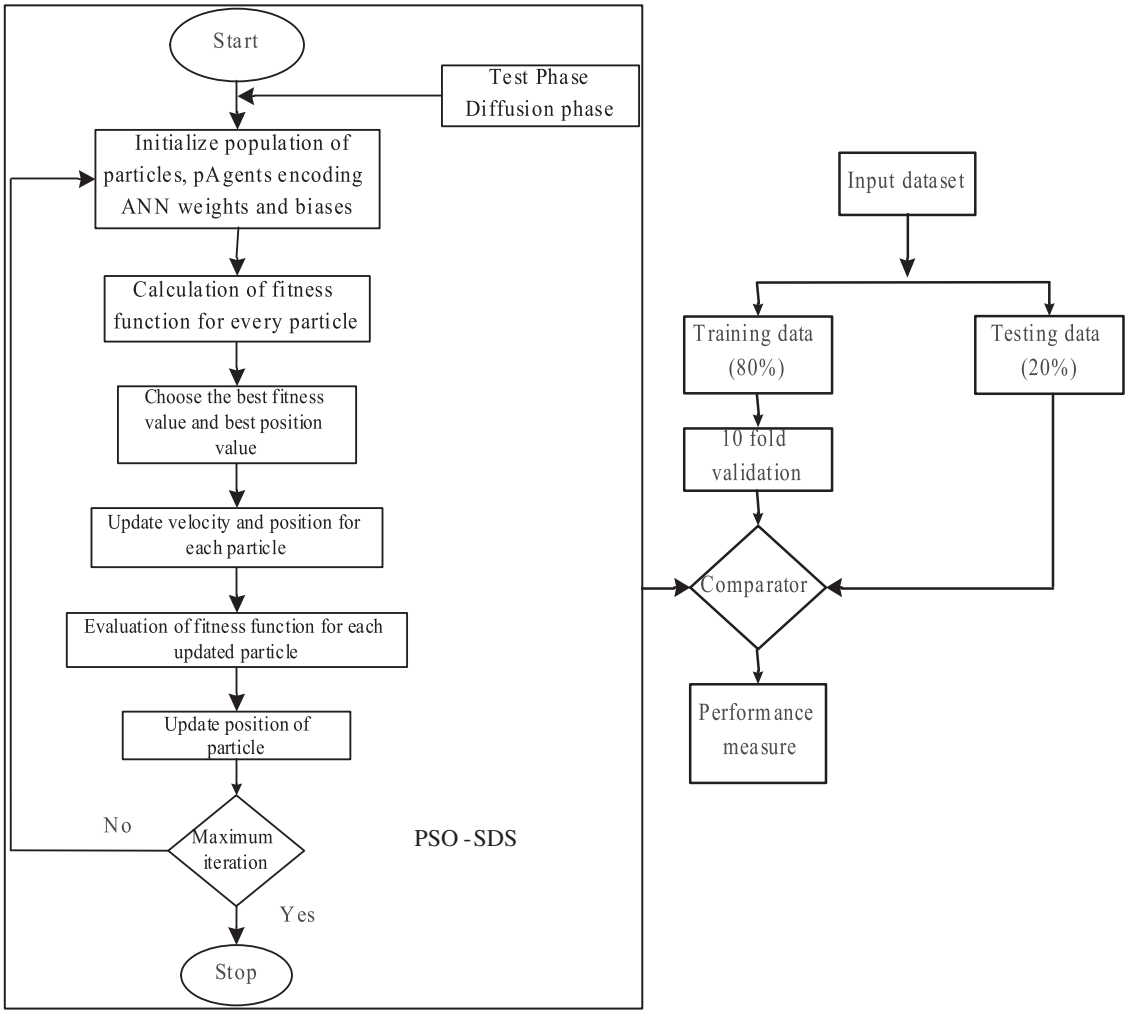

Figure 1: Overview of the proposed model

3.3 Particle Swarm Optimization (PSO)

The Particle Swarm Optimization (PSO) is a global optimization population-based approach that utilizes the heuristic search. John Kennedy and Eberhert devised the PSO in 1995. The origins of the PSO have come from computer science, engineering, social psychology, and artificial life. The PSO makes use of a "population" of particles with given velocities which traverse a problem hyperspace. Every object's speed is stochastically adjusted as per the neighborhood's optimal position and the historical optimal location for the particle itself, at each iterative step. A user-defined fitness function is used to derive the neighborhood best and the particle best. Each particle's movement will naturally evolve into a solution that is near-optimal or optimal. The particle's erratic movements in the problem space are referred to as the "swarm" which is more akin to a swarm of mosquitoes instead of a school of fish or a flock of birds. Being a computational intelligence-based method, the PSO is not majorly influenced by the problem's non-linearity and size. Unlike other analytical approaches, the PSO converges to the optimal response in numerous problems.

For every object, the fitness function is executed. The corresponding fitness value (which is also known as the best solution) is evaluated and kept for future use. Pbest is defined as the current optimum fitness value. lbest

is defined as the best population attained to date by any particle within the neighbors of the same location when the PSO optimization is done. The velocity and position for each generation are updated as per Eq. (1):

$$V_{pd}^{k+1} = \omega V_{pd}^{k} + c_1 r_1 (pbest^{k} - X_{pd}^{k}) + c_2 r_2 (gbest^{k} - X_{id}^{k})$$
$$X_{pd}^{k+1} = X_{id}^{k} + V_{pd}^{k+1} \quad (1)$$

Where, inertia weight (ranging from 0.2 to 0.9) is denoted by ω; iteration number is denoted by k; velocity found in the p-th object's d-th dimension is denoted by V^{k}_{pd}; actual position of the p-th object's d-th dimension is denoted by X^{k}_{id}; the particle's memory is denoted by pbest and gbest; the cognition and social factor are denoted by C_1 and C_2; the random functions consistently distributed in [0 1] are represented by r_1 and r_2.

3.4 Harmony Search (HS) Algorithm

The HS algorithm is based on how a musician seeks a better harmony for his musical performance, like jazz improvisation [27]. The HS algorithm will detect a different and pleasant harmony, which is decided through an aesthetic standard. Each musical instrument also has a pitch that is used for determining the aesthetic quality. The HS algorithm has benefits such as the minimal requirement for computations, and the lack of initial setting values required for the use of numerous decision variables. Derivative details is also not required as the HS approach utilizes a novel stochastic random search. When the current vectors are considered, the latest vectors are generated at the earliest.

The five steps which comprise the HS algorithm for the optimization methodology are as given below:

Step 1: The Initialization of Parameters – The detected problem in optimization is depicted as per Eq. (2):

$$\text{Minimize} f(x)$$
$$\textit{Subject to} \quad (2)$$
$$x_j \in X_j \; j = 1, 2, \ldots.., N$$

Here, the objection task is represented by f(x); a set deciding variable xj is indicated by x; the actual number of such selection variables is indicated by N; Every selection variable's another new set of the value range is denoted by Xj that is also a decision variable; a j-th decision parameter's upper and lower limits are denoted by $x_j^{\max}$ and $x_j^{\min}$.

Parameters of the HS algorithm comprise of: Stopping condition or the count of Improvisations (NI) where the solution vectors are stored, Pitch Adjusting Rate (PAR), Bandwidth Distance (BW), Harmony Memory Considering Rate (HMCR), and Harmony Memory Size (HMS) which is the solution vectors in a harmony memory.

Step 2: Harmony Memory Initialization and Evaluation identified is a novel random initial population which is produced as per Eq. (3):

$$x_{i,j}^{0} = x_j^{\min} + r_j (x_j^{\max} - x_j^{\min}) \quad (3)$$

Where i = 1, 2,…,HMS; j = 1,2,…,N and $r_j \in [0,1]$ will indicate an arbitrary number produced for each uniformly distributed j value. Accordingly, the analysis of the HM solution vectors is done, and the objective functions are evaluated.

Step 3: Improvisation – A new harmony vector is generated depending on the three rules: random selection, pitch adjustment, and consideration of memory. A design variable's value is selected from the HM's stored values with a probability that is not beyond the HMCR. The adjustment has been made together with the Pitch Adjusting Rate's probability, in which the randomly selected candidate values do not have to be considered with the other HM stored values with a probability of (1 - HMCR).

Step 4: The Harmony Memory –a New Harmony vector is identified and compared with a vector that is the worst in violation of the restriction where the poor vector is changed by the new vector.

Step 5: The checks for termination criterion – If the stopping criterion (also known as maximum improvisation) has been fulfilled, the HS is brought to an end. Otherwise, there will be a repletion of Step 3 and Step 4.

3.5 Artificial Neural Network (ANN) Classifier

ANN is a tool used for computational analysis. The ANN was influenced by the biological nervous system's functioning. Akin to how the brain's interconnected neurons receive, stores, recalls, analyses the data, and provide information for performance, ANN is also able to perform data processing through parallel computations. As the ANN is capable of learning from its earlier experiences, which are stored as data and also accurately execute the corresponding response patterns, researchers are very keen on utilizing it to resolve numerous clinical issues.

Artificial neural network models attain the biological neural networks' organization and functionalities. Artificial neurons are the fundamental building units of ANN. At an artificial neuron's entrance, the inputs are weighted by the multiplication of each input with a weight. All weighted neurons and bias are summed up by a function at the internal neuron. At the output neuron, the summation of the earlier weighted contributions and bias are transferred over an activation function. The products of an ANN with K element is provided as per Eq. (4):

$$y(x) = \sum_{i=1}^{k} w_i y_i(x) \tag{4}$$

Where y_i is the output at layer i and W_i is the weight at layer i.

Artificial Neural Network (ANNs) comprises of neurons, which are a collection of interlinked information processing units. The layers namely input, output, and single or multiple hidden layers, generally make up the neuron's three layers. Through synaptic weights, the input layer neurons will transfer the input signal to the initial hidden layer. A weighted sum of the inputs is evaluated by the hidden layer neurons. The activation function is also employed in order to decide if the value should be transferred to the subsequent layer. Hence, through weights adjustment, the neural network will carry on with its learning procedure.

3.6 Proposed PSO-SDS Algorithm

Every particle of the PSO found in a hybrid algorithm will have a memory, a velocity (which is a personal best position), and also a current position. However, every SDS agent has a hypothesis and status. The SDS is initialized with a set of randomly generated weights and biases assigned to the agents. Each agent is assigned to a combination of weights and biases (*i.e.*, hypothesis) from the search space.

The search space represents all possible combinations of weights and biases. This will be assessed partially by every agent found in the test phase of the SDS. Preserving a promising level of hypothesis can remove all unpromising ones, which will be its guiding heuristic. There have been several other tests that were done in the hybrid PSO-SDS algorithm for establishing the activity of every pAgent. The fitness of that of a random pAgent has been compared to the personal best of every particle within the test phase. The selected pAgent will then be flagged to be active with a good level of fitness value; if not, it is flagged as inactive. This type of methodology will ensure that about 50% of the pAgents, on an average, are available for every step in the iteration.

Every inactive pAgent randomly chooses one more pAgent in its Diffusion Phase. Here, in this phase, in case there is an active pAgent that is chosen, the selected pAgent and its hypothesis will be conveyed to the pAgent that is inactive. If the inactive pAgent was chosen, there was a novel hypothesis that is generated randomly from its search space by choosing the pAgent. Another complete SDS cycle has been performed after each n number of function evaluations. Such hybrid algorithms are called the SDS-PSO, wherein there are many PSO evaluations that are performed before the SDS cycle and are denoted by r. The hybrid algorithms and its behaviour has been shown in this algorithm (as depicted in Fig. 2).

```
Harmony Search PSO-SDS Algorithm:
Configure pAgents
Until (stopping formula is not met)
For every pAgents
        Fitness value is evaluated for each particle
                If (counter MOD n=0)
                //START SDS
                //TEST PHASE
                   for agent=1 to No_of_pAgents
                   r_agent=pick-random-pAgent ()
                         if (agent.pbestFitness () <= r_agent.pbestFitness ())
                            agent.setActivity (true)
                         else
                            agent.setActivity (false)
                         end if
                   end for
                //DIFFUSION PHASE
                   for agent=1 to No_of_pAgents
                         if (agent.activity () == false)
                         r_agent=pick-random-pAgent ()
                            if (r_agent.activity () == true)
                               agent.setHypo (r_agent.getHypo ())*
                            else
                               agent.setHypo (randomHypo ())
                            end if
                         end if
                     end for
                //End SDS
```

Figure 2: Pseudocode for proposed PSO-SDS algorithm

3.7 Proposed Harmony Search with SDS Algorithm

The Harmony Memory (HM) is required for storing specific feasible vectors within a feasible space for HS algorithm utilization. A musician uses three distinct rules to improve a specific pitch: i) generating an individual pitch taken from memory which denotes a single value's selection from the harmony memory is known as consideration of memory; ii) generating another pitch with memory by selecting a harmony memory's adjacent value is known as pitch adjustments; iii) randomization is the generation of a random pitch from a value range. In the HS algorithm, any one of these rules is employed when a decision variable has to select an HS value. Within the memory, when another new harmony vector becomes available, this new vector will cause an alteration to it. Until the stopping criterion is fulfilled, this methodology is continuously repeated.

4 Results and Discussion

Software test automation has been supported by automated software test oracles. Automated software test oracle is not facilitated by many automated software testing tools including the open-source tools, Selenium and cucumber. Implementation of the automated software test oracle was carried out in this

work. MATLAB and customized WEKA tool were used for the simulation of the test oracle. The network is trained by injected faults dynamically is carried out. Comparison were made between the new faults injected (single and multiple) with the testing data and the misclassification rate is calculated. The Tab. 1 provides the injected faults.

Table 1: Faults injected

Statement originally in the software	Injected faults
To check whether 2 argument is present using OR operator	Operator is Removed to show only one
OR Operator used	Restored OR Operator with AND operator
> Operator Used	Restored with > Operator
> Operator Used	Restored with = Operator
> Operator Used	Restored with < operator
To check 2 criteria, AND Operator used	Restored with OR operator
Argument change	Numerical value is used

The Misclassification of correct output of the proposed approach is specified in Tab. 2 in accordance with the faults injected.

Table 2: MCO for Proposed Harmony Search-SDS NN

Faults injected	PSO-SDS NN	Harmony Search-SDS NN
1	3.03	2.91
2	3.02	2.92
3	5.4	5.22
4	5.99	5.75
5	6.63	6.37

The MCO values of the proposed and other approaches for the iterations 1 to 5 are depicted in Fig. 3. It is clearly evident that the proposed HS SDS NN approach outperforms all other approaches irrespective of the number of iterations used. The MOC of the Harmony Search-SDS NN operates better by 4.04%, 3.37%, 3.39%, 4.09% and 4% than Proposed PSO-SDS Neural Network at 1, 2, 3, 4 and 5 iteration respectively.

The performance of proposed approach in terms of MWO is given in Tab. 3. The performance of all the approach is almost similar in with iteration 1. As the number of iterations increases and MWO of ANN is 13.3 which comparatively high by 4.9 for PSO-SDS NN and by 5.15 for HS SDS NN scheme. Fig. 4 reveals that the MWO of Proposed Harmony Search-SDS NN operates better by 3.15%, 3.9%, 3.42%, 3.41% and 3.5% than Proposed PSO-SDS NN at 1, 2, 3, 4 and 5 iteration respectively. The MWO of HS SDS NN is better than all other approach due to the usage of harmony search algorithm.

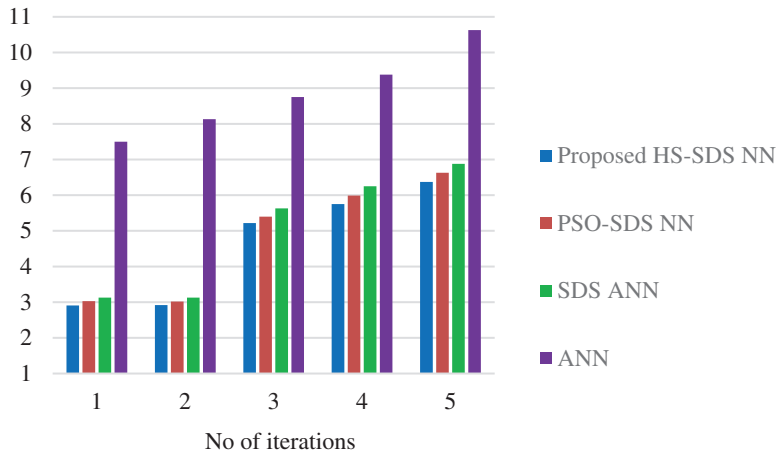

Figure 3: Performance comparison of MCO

Table 3: MWO of proposed approach

Faults injected	PSO-SDS NN	Harmony Search-SDS NN
1	6.12	5.93
2	6.8	6.54
3	7.73	7.47
4	8.36	8.08
5	8.43	8.14

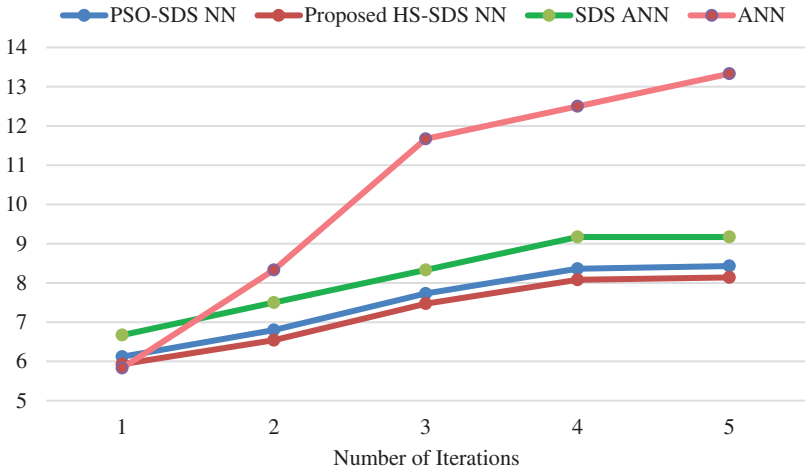

Figure 4: Performance measure of MWO with other approaches

Fig. 5 shows that the CO of the proposed Harmony Search-SDS NN occurred at 500th iteration, with a value of 0.23. Also, the convergence of the proposed PSO-SDS NN occurred at iteration number 550, with a value of 0.36.

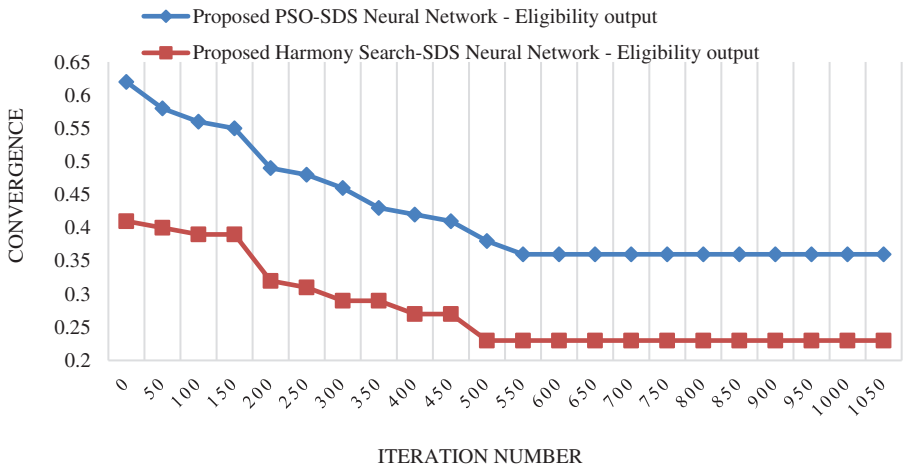

Figure 5: CO for proposed harmony search-SDS neural network

An artificially manufactured data was created instead of the real time events. Synthetic data is created programmatically, to train the machine learning model and it is used as a stand-in for test datasets of production. The advantage of using synthetic information is to ensure the confidentiality of the data. Tabs. 4 and 5 shows the outcome for misclassification of correct output, misclassification of wrong output. Figs. 6–8 shows the results for misclassification of correct output, misclassification of wrong output and convergence, respectively.

Table 4: MCO for proposed harmony search-SDS NN for synthetic dataset

Faults injected	PSO-SDS NN	Harmony Search-SDS NN
1	3.2	3.09
2	3.18	3.05
3	5.62	5.42
4	6.36	6.11
5	6.96	6.71

Table 5: MWO for proposed harmony search-SDS NN for synthetic dataset

Faults injected	PSO-SDS NN	Harmony Search-SDS NN
1	6.67	6.46
2	7.59	7.35
3	8.35	8.08
4	9.29	8.97
5	9.2	8.89

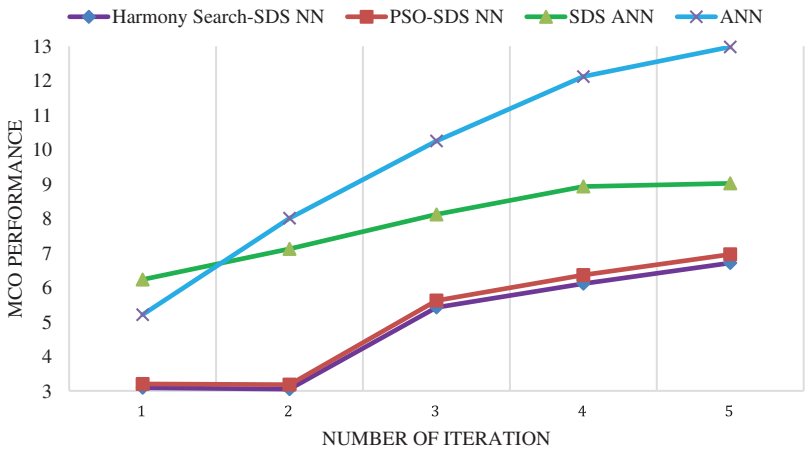

Figure 6: MCO Comparison for Synthetic dataset

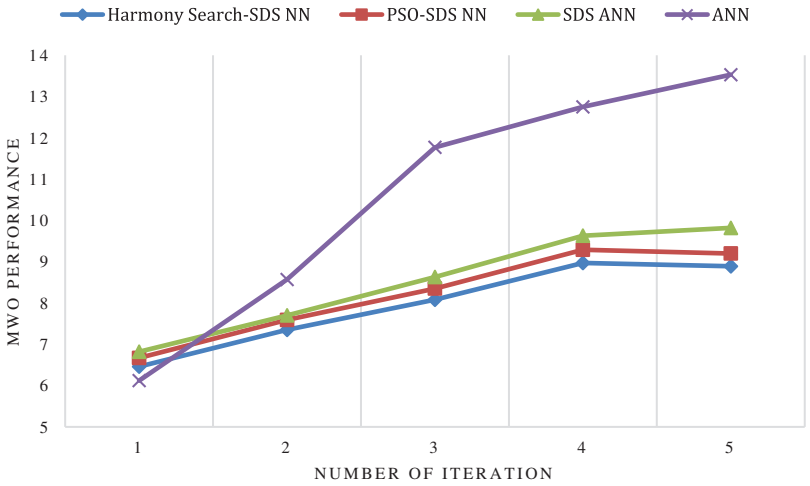

Figure 7: MWO comparison for synthetic dataset

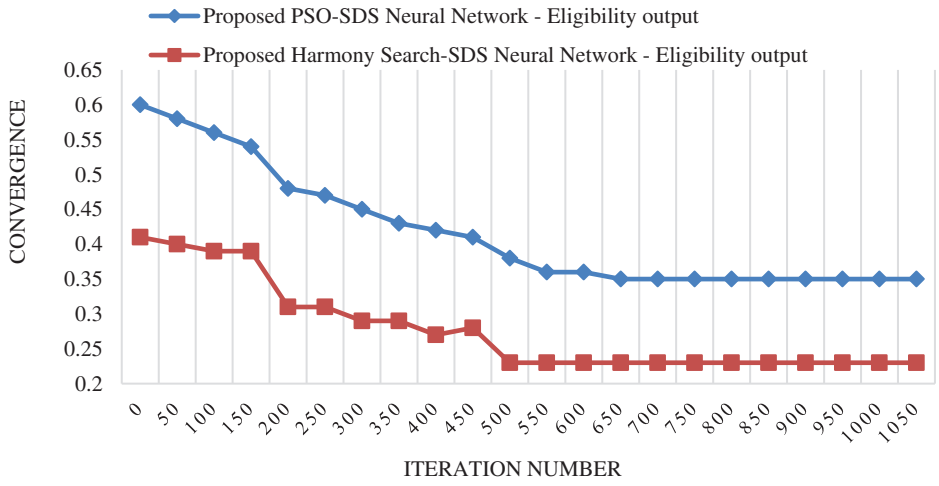

Figure 8: CO for proposed harmony search-SDS NN

Fig. 6 shows that the MCO of Proposed Harmony Search-SDS NN performs better by 3.5%, by 4.17%, by 3.62%, by 4% and by 3.66% than PSO-SDS NN 1, 2, 3, 4 and 5 iteration respectively. It is clearly evident that Harmony Search-SDS NN is superior than PSO- SDS NN. The MCO of PSOSDS NN and HS-SDS NN are almost same in all the iterations and HS-SDS NN approach performs better the PSO SDS approach by a small margin in all cases.

Fig. 7 reveals that the MWO of Proposed Harmony Search-SDS NN functions better by 3.19%, 3.2%, 3.29%, 3.5% and 3.43% than Proposed PSO-SDS NN at number 1, 2, 3, 4 and 5 iteration respectively. For all these iterations the performance of HS-SDS NN is superior than PSO SDS NN method. Even though the MWO of ANN performs is iteration 1 but as the number of injected fault increases the performance of ANN decreases and MWO is 13.53% for iteration 5. In case of proposed HS-SDS NN approach the MWO increases with number of fault but still it performs compared to other approaches.

Fig. 8 shows that the CO of the Proposed Harmony Search-SDS NN occurred at 500th iteration, with a value of 0.23. Also, the convergence of the Proposed PSO-SDS NN occurred at 650th iteration, with a value of 0.35.

5 Conclusion

Errors can result in software failure. Software testing is carried to detect these faults. A regression test suite is used for the test cases selection, to maximize the test case's specific objectives that aid in decreasing the cost and time required for the service-oriented business application maintenance. Experimental results for the real dataset demonstrate that the MCO of the Proposed Harmony Search-SDS NN does better compared to the Proposed PSO-SDS NN by 4.04% at iteration number 1, by 3.37% at iteration number 2, by 3.39% at iteration number 3, by 4.09% at iteration number 4, and by 4% at iteration number 5. Experimental results for the Synthetic dataset also demonstrate that the MCO of the Proposed Harmony Search-SDS NN does better compared to the Proposed PSO-SDS NN by 3.5% at iteration number 1, by 4.17% at iteration number 2, by 3.62% at iteration number 3, by 4% at iteration number 4, and by 3.66% at iteration number 5. Among the two approaches proposed the HS-SDS NN performs better than PSO-SDS NN approach and is well suited for automated testing. Further Gravitational Search Algorithm can be utilized for enhancing the performance and can be evaluated for large scale software program in a real time environment.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. Agarwal, D. E. Tamir, M. Last and A. Kandel, "A comparative study of artificial neural networks and info-fuzzy networks as automated oracles in software testing," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 45, no. 5, pp. 1183–1193, 2012.

[2] E. T. Barr, M. Harman, P. Mcminn, M. Shahbaz and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 4, no. 5, pp. 507–525, 2015.

[3] R. A. Oliveira, U. Kanewala and P. A. Nardi, "Automated test oracles: State of the art, taxonomies, and trends," *Advances in Computers*, vol. 95, no. 4, pp. 113–199, 2015.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of ICNN'95 - Int. Conf. on Neural Networks*, Perth, Australia, vol. 4, pp. 1942–1948, 1995.

[5] S. Das, P. K. Singh, S. Bhowmik, R. Sarkar and M. Nasipuri, "A harmony search based wrapper feature selection method for holistic bangla word recognition," *Procedia Computer Science*, vol. 89, no. 3, pp. 395–403, 2016.

[6] S. R. Shahamiri, W. M. N. Wan-Kadir and S. Z. Mohd-Hashim, "A comparative study on automated software test oracle methods," in *Fourth Int. Conf. on Software Engineering Advances*, Porto, Portugal, pp. 140–145, 2009.

[7] K. Kamaraj and C. Arvind, "Strategies of automated test oracle – A survey," *Advances in Natural and Applied Sciences*, vol. 11, pp. 1998, 2017.

[8] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2014.

[9] K. Senthil Kumar and A. Muthukumaravel, "A hybrid approach for test case prioritization using pso based on software quality metrics," *International Journal of Engineering & Technology*, vol. 7, no. 3.12, pp. 300–303, 2018.

[10] R. Braga, B. S. Neto, R. A. Rabelo, J. Santiago and M. Souza, "A machine learning approach to generate test oracles," in *SBES '18: Proc. of the XXXII Brazilian Sym. on Software Engineering*, Sao Carlos, Brazil, pp. 148–151, 2018.

[11] S. R. Shahamiri, W. M. Wan-Kadir, S. Ibrahim and S. Z. Hashim, "Artificial neural networks as multi-networks automated test oracle," *Automated Software Engineering*, vol. 19, no. 3, pp. 303–334, 2011.

[12] M. E. Yousif, S. R. Shahamiri and M. B. Mustafa, "Test oracles based on artificial neural networks and info fuzzy networks: A comparative study," in *2015 IEEE 10th Conf. on Industrial Electronics and Applications (ICIEA)*, Auckland, New Zealand, pp. 467–471, 2015.

[13] S. R. Shahamiri, W. M. N. Wan-Kadir, S. Ibrahim and S. Z. Mohd Hashim, "An automated framework for software test oracle," *Information and Software Technology*, vol. 53, pp. 774–788, 2011.

[14] K. L. Du and M. N. S. Swamy, "Particle swarm optimization," in *Search and optimization by metaheuristics*, Birkhäuser, Cham, pp. 153–173, 2016.

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

[16] W. Xu, T. Ding, H. Wang and D. Xu, "Mining test oracles for test inputs generated from java bytecode," in *IEEE 37th Annual Computer Software and Applications Conference*, Kyoto, Japan, pp. 27–32, 2013.

[17] H. A. Alhakbani and R. M. Al-Rifaie, "Feature selection using stochastic diffusion search," in *Proc. of the Genetic and Evolutionary Computation Conf.*, Berlin, Germany, pp. 385–392, 2017.

[18] A. Singhal Vineeta and A. Bansal, "Generation of test oracles using neural network and decision tree model," in *5th Int. Conf. - Confluence The Next Generation Information Technology Summit (Confluence)*, Uttar Pradesh, India, pp. 313–318, 2014.

[19] X. S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm. Studies in Computational Intelligence*, Berlin, Heidelberg, Springer, vol. 191, pp. 1–14, 2009.

[20] I. Hooda and R. S. Chhillar, "Test case optimization and redundancy reduction using GA and neural networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 5449, 2018.

[21] O. Sangwan, P. Bhatia and Y. Singh, "Radial basis function neural network based approach to test oracle," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 5, pp. 1–5, 2011.

[22] J. M. Altmemi, R. R. Othman, R. Ahmad and A. S. Ali, "Implementation of sine cosine algorithm (SCA) for combinatorial testing," *IOP Conference Series: Materials Science and Engineering*, vol. 767, no. 1, pp. 012009, 2020.

[23] S. Liu and S. Nakajima, "Automatic test case and test oracle generation based on functional scenarios in formal specifications for conformance testing," *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 691–712, 2022.

[24] R. Zhang, Y. W. Wang and M. Z. Zhang, "Automatic test oracle based on probabilistic neural networks," *Recent Developments in Intelligent Computing, Communication and Devices. Advances in Intelligent Systems and Computing*, vol. 752, pp. 437–445, 2019.

[25] K. Kamaraj, C. Arvind and K. Srihari, "A weight optimized artificial neural network for automated software test oracle," *Soft Computing*, vol. 24, no. 17, pp. 13501–13511, 2020.

[26] M. M. Al-Rifaie, J. M. Bishop and T. M. Blackwell, "An investigation into the merger of stochastic diffusion search and particle swarm optimisation," in *GECCO '11: Proc. of the 13th Annual Conf. on Genetic and Evolutionary Computation*, Dublin, Ireland, pp. 37–44, 2011.

[27] Z. W. Geem, "State-of-the-Art in the structure of harmony search algorithm," in *Recent Advances In Harmony Search Algorithm. Studies in Computational Intelligence*, Berlin, Heidelberg, Springer, vol. 270, 2010.