Enhancing Test Suite Optimization in Software Engineering through a Hybrid Whale Optimization and LSTM Approach

Shuqing Qiao
College of computer science and cyber security (pilot software college), Chengdu University of Technology, Chengdu Sichuan,610059, China
E-mail: 15386513158@163.com

Keywords: test suite, software engineering, whale optimization algorithm, test generation, parameters tuning

Recieved: June 10, 2025

The algorithms used for Meta are commonly utilized to solve complicated optimization issues, but they frequently have disadvantages such as premature convergence, poor examination in the initial phases of the search, and insufficient adaptation of search operators. To address these issues, this study presents a novel hybrid approach, the Whale Optimization Algorithm with Long Short-Term Memory (WOA-LSTM), which is intended to improve exploration and adaptive learning. The primary goal is to overcome the performance limits of traditional WOA by providing a more robust search approach for confined combinatorial test-generating tasks. The proposed WOA-LSTM combines the regular WOA with an LSTM-based probability table that dynamically selects search operators. Data normalization with Z-score standardization resulted in stable feature scaling for effective LSTM training. In the early iteration stages, low-performing operators are given a chance to re-enter the search, preventing premature stagnation. Furthermore, the model selects just-in-time adaptive operators among the three WOA mechanisms random generation, spiral shape, and shrinkage—based on their historical effectiveness, allowing for a balance of exploration and exploitation. The experimental results show that WOA-LSTM outperforms other algorithms, with an accuracy of 99.3% and a recall of 99.1%. Comparative examination shows that the suggested method outperforms traditional WOA while being competitive with other sophisticated metaheuristic algorithms. Furthermore, statistical validation with t-tests demonstrates WOA-LSTM's considerable superiority over existing techniques, ensuring the proposed model's performance is reliable, resilient, and generalizable across several experimental runs. Finally, WOA-LSTM presents a highly successful optimization framework that overcomes WOA's drawbacks while also pointing the way forward for future optimization research.

Povzetek: Članek obravnava neučinkovitosti klasičnega algoritma kita (WOA) pri optimizaciji testnih naborov, predvsem prezgodnjo konvergenco in šibko zgodnjo eksploracijo. Predlaga hibrid WOA-LSTM, kjer LSTM prek verjetnostne tabele sproti izbira najboljše WOA-operatroje. Metoda izboljša ravnovesje eksploracije/eksploatacije.

1 Introduction

The research suggests a novel method for t-way (TW)test suite (TS) development that supports restrictions based on the Whale Optimization Algorithm (WOA), was created by taking inspiration from the humpback whale's hunting habits. WOA's unique optimization mechanism gives it a powerful global search capability [1]. Furthermore, WOA has a simple implementation and is less dependent on parameters. As a result, it is often proposed to solve a varied range of problems, such as feature selection, energy, electrical power, flow shop scheduling, clustering, and electronic engineering. Furthermore, the WOA has revealed modest results in every field [2]. Given that WOA performs well against a variety of meta-heuristics, its usage for the presently recommended combinatorial TW TS development is warranted [3].

The literature has developed a number of useful metaheuristic-based methods that consider TW testing as an optimization difficult. The performance of each metaheuristic-based TW technique is often determined by its backbone algorithm [4]. To reevaluate the worst-performing operator in the initial iteration and investigate additional search regions, the Exponential Monte Carlo algorithm's (EMC) acceptance probability is also incorporated. Testing is a crucial step in the software (Sof) development life-cycle and is part of the quality assurance effort. Extensive testing is nearly impossible due to budget and time constraints, even though it is desirable to guarantee adherence to specifications [5]. In the literature, numerous sampling-based strategies (such as boundary

values analysis and equivalence partitioning) have been put forth to minimize testing [6]. Despite their usefulness, a large number of current sampling techniques do not account for interaction-related faults (also known as TW testing, where t denotes the degree of the interaction).

Finding a explanation that satisfies all or maximum of the given restrictions while elevating a certain performance metric under a set of predetermined situations is the fundamental goal of optimization issues [7]. Optimization difficulties have become much more complicated and diverse as a result of the rapid breakthroughs in science and technology [8]. These difficulties, which frequently include high-dimensional spaces, go beyond conventional linear or quadratic programming and include characteristics like, dynamic changes, nonlinearity, multi-impartiality, and high uncertainty [9]. Researchers and engineers have created a wide variety of optimization procedures to successfully handle these crucial and more complicated optimization problems [10]. Particularly, swarm intelligence algorithms have become well-known because of their distinct benefits and proven ability to solve challenging optimization issues [11].

Contribution of the study:

- The study is the first of its type to use WOA for the creation of TW TS.
- Using a series of benchmark TS, the study thoroughly assesses WOA's performance.
- Next, just-in-time adaptive selection of the best-performing operators (between random generation, spiral shape mechanism, and shrinking mechanism) is made possible by a new WOA-LSTM variant algorithm at any given iteration. To avoid local optima, the study uses LSTM.
- The proposed study provides the effective results of the combinatorial TS with constraint support.

2 Literature review

The researcher presented GWOATEO, a novel hybrid optimisation strategy that combines heat exchange optimisation methods and the genetic algorithm in a synergistic manner [12]. By employing memory-based limit methods and location informing techniques for its leading answers, this algorithm improves its search capabilities. An antagonistic learning approach was also used in the study to produce the opposite of the global optimum solution. This improved the search efficiency and procedure performance. The multi-leader guided multi-objective WOA that includes an oppositional learning technique (OLT) to enhance the initial population spreading. The OLT is successful in generating a contrasting answer to the global optimum, hence improving the overall optimal solution, in addition to optimising the initial population distribution [13]. The researcher balanced algorithmic exploration and exploitation by employing an elite search strategy and an adaptive variable-speed technique to develop global optimisation [14]. They successfully higher the trade-off between algorithmic advancement and search performance by employing adaptive inertia weights, modified global optima, and historical individual optimal solutions. The related a WOA that integrates the Lévy flight trajectory mechanism to enhance the algorithm's capacity to avoid local optima, avoid premature convergence, and encourage population variety. The researcher enhanced the WOA by incorporating a limited mutation mechanism to increase convergence efficiency and adaptive nonlinear inertia weights to control convergence speed in order to address economic load scheduling problems [15]. For the first population generation and convergence stages, the researcher used a dynamic penalty function method using the Gaussian Variational methodology. These improvements greatly increased the algorithm's accuracy, resilience, stability, convergence rate, and global search capabilities. Furthermore, they demonstrated advantageous outcomes when dealing with increasingly intricate constrained optimisation issues. In order to balance exploration and exploitation, avoid stagnation, and enhance the quality of initial solutions, the researcher employed scheduling concepts, nonlinear convergence elements, and mutation approaches. The energy-saving scheduling issue has been successfully resolved by these approaches.

Table 1 displays the summary of recent studies on TS optimization utilizing various algorithms, with emphasis on aims, applicable methodologies, and constraints, particularly in terms of performance and scalability of WOA versions.

Table 1: Established from current research on metaheuristic-based TS optimization.

References	**Objectives**	**Methods**	**Limitation**
Deneke et al., [16]	To reduce regression TS, remove unnecessary test cases (TC), improve productivity, and lower execution costs in Sof testing.	Particle Swarm Optimization (PSO) was used to reduce TS size in comparison to Greedy Weighted Set Cover (GWSC), etc.	Performance is dependent on dataset features; PSO necessitates careful parameter adjustment and may not always result in globally optimal TS reduction.
Ahmed et al., [17]	Create a prioritized TW TS generating approach that balances TC weight and priority for effective Sof testing.	The Bi-objective Dragonfly Algorithm (BDA) creates prioritized TW TS.	BDA could demand more processing resources for large datasets, and prioritizing relies on proper weight and priority assignment.
Zayed and Maashi [18]	To save effort, time, and money on regression testing, minimize the size of the TS while increasing Sof coverage.	Genetic and greedy algorithms optimize regression TS efficiently.	The method could struggle with large Sof systems. Solution quality was dependent on algorithm parameters and initial TS design.
Ouyang et al.,[19]	To address the TS case issues, the Improved WOA (ImWOA) is presented in this study.	Improved WOA	The application of the whale optimization technique to large-scale TS optimization problems may be limited by its high computer resource requirements.
Zamani et al., [20]	The limitations of WOA in resolving global optimisation issues, including its propensity to converge slowly and its susceptibility to local optimum, are addressed in this study by proposing MSWOA.	MSWOA	The complexity and features of the TS may have an impact on how well the whale optimization technique performs.
Abdollahzadeh et al., [21]	With an emphasis on both traditional and cutting-edge techniques, such as deep learning, hybrid approaches, and generative AI models, the study highlights current trends, difficulties, and prospects for additional research.	TS optimization	It may be necessary to carefully adjust the whale optimization algorithm's settings, which can take time, to maximize its performance.
Gharehchopogh et al., [22]	This study examines the various tools, regression TS optimization approaches, and mathematical models currently in use for RTO.	Genetic algorithm	A tiny TS and the genetic algorithm's population size won't provide the GA with adequate solution space to generate reliable findings.
Cao et al., [23]	This study looked into a method for saving time and effort when performing regression testing.	meta-heuristic algorithm	WOA experiences premature convergence, much like any other meta-heuristic algorithm, including swarm and evolutionary algorithms.
Wang et al., [24]	This study examines the various tools, regression TS optimization approaches, and mathematical models currently in use for RTO.	Regression Test Suite Optimization (RTO)	The study might not offer a thorough comparison with other optimisation methods, which could restrict our comprehension of the relative performance of the whale optimisation technique.

Liu et al., [25]	In this paper, we propose a four-step reinforcement learning-based TS prioritization methodology.	Support vector machine (SVM)	Examining the WOA scalability for extensive TS optimization issues.
Burachynskyi and Shantyr [26]	Analyze AI integration in Sof development to improve processes, automate jobs, increase product quality, lower costs, improve testing, streamline project management, and enable adaptive, intelligent Sof systems.	Literature review, case studies, AI model analysis, workflow assessment.	The research was founded on previous research; practical implementation hurdles, dataset limits, model generalization issues, and diversity in organizational adoption may all have an impact on the applicability of AI techniques across various Sof development settings.

2.1 Research gap

Due to its simple design, strong adaptability, and few control parameters, researchers have found the WOA to be especially interesting. Numerous complicated optimisation problems, including fuzzy programming, DNA fragment assembly, feature selection and workshop scheduling, image segmentation, engineering design, and path planning, have been successfully solved by this technique [12]. However, when applied to complex issues and high-dimensional data, the WOA's shortcomings become evident. In particular, its optimisation pace is somewhat poor, frequently falling short of the need for quick fixes. Furthermore, the optimisation accuracy is too low, which could result in less-than-ideal solution quality. Additionally, the WOA's search and exploitation capabilities are lacking, which limits its efficacy in real-world applications by impeding its capacity to fully and profoundly explore the solution space. Many researchers have carried out in-depth analyses and put forth a range of successful enhancement strategies in an effort to overcome these limitations and improve the WOA's performance. By improving the WOA's algorithmic framework with deep learning techniques, parameter configurations, search tactics, and other pertinent elements, these techniques aim to maximise and improve the solution results.

3 Materials and methods

The research uses WOA- LSTM to optimize TS. The preprocessing step includes data cleaning and applying Z-score standardization, which essentially transforms the data values into a stable input. The WOA mimics the feeding behaviors of whales with encircling, spiral, and random operators that are all inherently guided by a probability table. The LSTM captures dependencies of sequential data points from the historical data inputs, allowing for accurate sequence generation. Once the LSTM and WOA are integrated together, outcomes observed are an efficient exploration versus exploitation, and generated and optimized TS with high performance. Figure 1 shows the methodology flow diagram for TS Optimization.

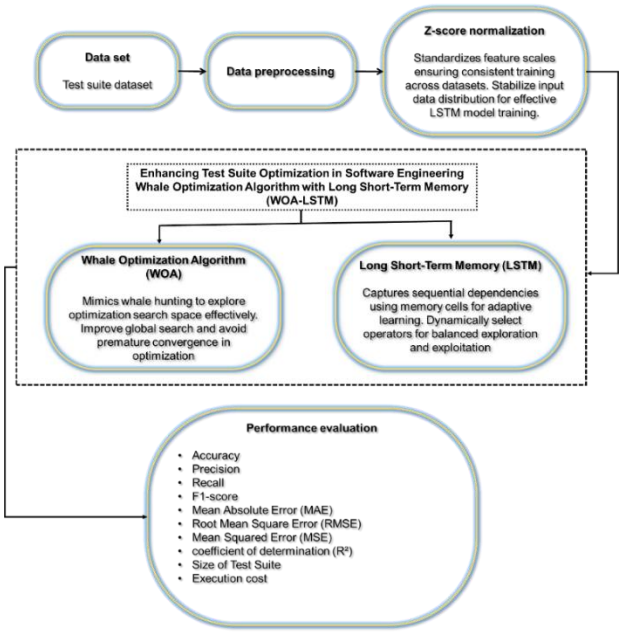

Figure 1: show the methodology flow diagram.

3.1 Data collection

A "test suite dataset" is a group of data used to assess how well an algorithm or model performs. It is a collection of scenarios, TC, or scripts used to verify the performance and functioning of a system or Sof program. The data is obtained from Kaggle https://www.kaggle.com/datasets/bwandowando/mushroom-overload. Kaggle's mushroom dataset has 8,124 instances with 22 categorical attributes that describe multiple mushroom features (e.g., cap shape, surface, color, gill size, odor, habitat). Each observation instance was either labeled edible or poisonous; therefore, the mushroom dataset is appropriate for a classification and optimization problem. The mushroom dataset is particularly useful in evaluating TS generation models because it has a variety of well-defined categorical data. The mushroom dataset also provides an appropriate level of complexity to allow the proposed WOA-LSTM to

demonstrate the robustness of feature handling, exploration, and adaptive learning abilities while generating a TC sequence.

3.2 Data preprocessing using Z-score normalization

Z-score normalization is a statistical method that normalizes input by removing the mean and obtaining unit variance.
With normalization, can guarantee that features contribute equally to the learning process and reduce the influence of extreme values or outliers. Z-score normalization maps values less than the mean on the negative scale, greater than the mean on the positive scale, and sets zero maps to zero. The goal of normalization is to establish standard and s **Parameter tuning** feature scales, leading to model fairness, reduced bias, and better optimization in optimization frameworks. During Z-score normalization, the transformation is as follows in equation (1).

$$u' = \frac{u-\mu}{\sigma} \quad (1)$$

u' Is the normalized feature, u is the original feature value, μ is the mean, and σ is the standard deviation. The process converts the dataset into a distribution with a mean of zero and a variance of one, resulting in a standardized environment for algorithmic training. The change simplifies feature comparison while also improving the interpretability of adaptive system outputs. By consistently rescaling data, the approach allows for methods such as LSTM to be trained with balanced inputs, preventing skewed outcomes. To verify that all features have a consistent statistical distribution, facilitating stable convergence and balanced operator assessment in WOA-LSTM.

3.3 T way

There is usually a strict timeline that the testers must follow. Therefore, very successful testing techniques are required to achieve optimal test coverage and a high rate of problem detection. TW testing provides the features mentioned above. TW testing is a technique that looks at every distinct combination of the Sof system's parameters. The Covering Array (CA) is a particular Design of Experiments, called a TS. Making a TC from a Sof system input or system configuration is how t-. The purpose of TW testing is to thoroughly investigate interactions between various input parameters in a Sof system, so that serious faults caused by parameter combinations are discovered early.

3.4 (DoE) strategies used in T-way testing

CAs generalize OAs by requiring that the t-tuples be included at least once rather than a predetermined number of times in order to obtain balanced t-tuples. A tuple is a finite, ordered list of mathematical elements, such as parameter interaction values. A t-tuple is an ordered set of t elements. The combinatorial object CA is represented as CA (N; t, vp), where t is the strength of the interaction, v is the parameter value, N is the number of test instances generated, and p is the parameter (e.g., input data, system configurations, or both).

The primary objective of TW testing is to minimise the size of the TS without sacrificing its ability to identify problems. The TW testing is to reduce the overall size of the TS while ensuring that all essential parameter interactions are covered, thus balancing efficiency and effectiveness. A group of TC designed to evaluate a Sof system and illustrate a certain set of behaviors, in a way that challenging is carried out. A variety of examination scenarios are then used to build the Sof system's TS in Figure 2.
Figure 2 illustration below depicts the step-by-step procedure of creating a combinatorial TS with the WOA-LSTM algorithm. That starts with a random selection of beginning TC from the t-tuple table. The WOA-LSTM algorithm is then used to optimize these TC, with a fitness function that reduces weight. The best TC with the highest number of interactions is chosen. The TC covers and removes the corresponding entries from the t-tuple table. The operation continues until the table is empty.

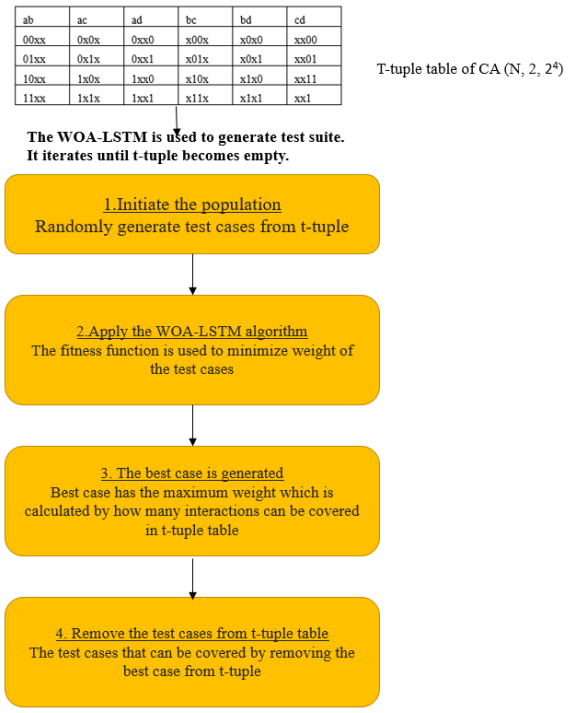

Figure 2: T way testing implementation

3.5 Whale optimization algorithm with long short-term memory (WOA-LSTM), used to improve exploration and adaptive learning

The combined use of LSTM and a WOA incorporates sequential learning with an adaptive search to optimize the TS in Sof testing. In these pairs of algorithms, LSTM captures temporal dependencies in a sequence while providing candidates of test sequences, while WOA evaluates and filters through candidates by using encircling, spiral, and random operators integrated with a probability table to dynamically adjust operator selection - remembering past performance to boost efficiency while balancing between exploration and exploitation. This combination addresses redundant TC, enhances fault detection, results in an optimized TS with high accuracy and reliability in testing, and is effective in handling the complexity and size of modern Sof systems.

The WOA is selected for its successful exploration and exploitation capability, but it suffers from early convergence and limited versatility. LSTM is incorporated to expand the capability of WOA, as it can represent and capture sequential performances and is adaptable to operator selection. The hybrid WOA-LSTM exploits the exploratory strength of WOA paired with the adaptability given from sequential memory by LSTM, which yields a balanced exploration and exploitation process. This displays a better convergence time, prevents stagnation, and ultimately provides a greater accuracy in a rich and complex combinatorial optimization problem, therefore offering a strong optimization framework.

Whale optimization algorithm: One metaheuristic optimisation technique that can be used for Sof engineering TS optimisation is the WOA. It is employed to identify the optimal subset of TC to minimise the number of TC conducted and maximise fault detection. By employing tactics like encircling prey and bubble-net attacking to investigate and take advantage of the search space, WOA mimics the hunting behaviour of humpback whales in Figure 3. WOA is a designer-inspired meta-heuristic based on the foraging behavior of humpback whales, which uses three operators - encircling prey, spiral bubble-net attacking, and random exploration - to find optimal solutions. In this research, WOA is applied to TS optimization, where the "prey" is the perfect subset of TC. By continuously updating the position of the whales, WOA will minimize redundant TC while maximizing coverage of fault detection, and this is done efficiently.

Figure 3 displays the WOA. The whale represents the leader directing the search, and the fish represent agents or candidate solutions. The spirals depict what whales might do during encircling and bubble-net attacking, representing the exploitation behaviour of the WOA algorithm. The clusters of spheres above represent distinct solution spaces. The swarm of fish or agents converges around possible high-quality solutions, representing how WOA explores and exploits the search space. The image represents both the exploration and convergence dynamics of WOA.

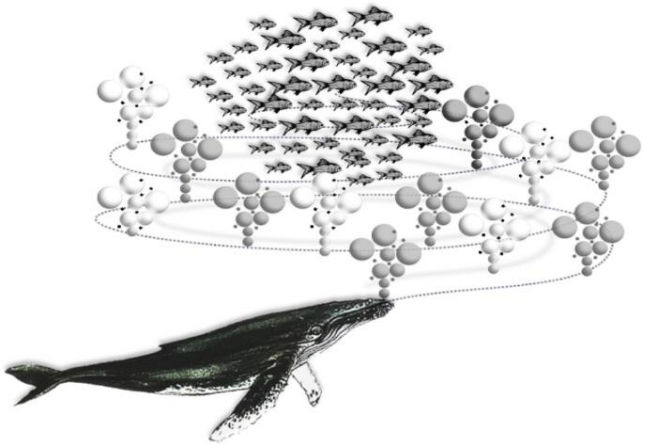

Figure 3: Whale optimization

Test suite optimisation makes use of WOA: WOA considers the ideal TS to be the "prey" in the search space, encircling it. By iteratively updating their positions and getting closer to the best-found answer, whales (search agents) efficiently cut down on the number of TC while preserving coverage.

Bubble-Net Attacking: WOA imitates humpback whales' spiral attack. This facilitates search space exploration, especially when looking for answers that are not immediately close to the best-found one.

Whales increase diversity and avoid premature convergence to local optima by randomly exploring the search space while looking for prey.

Encircling prey: Whales iteratively search for the optimal solution. Bubble-Net Attacking: WOA mimics humpback whales' spiral attacks to explore local/global areas. Random exploration: Whales diversify their searches to avoid premature convergence. In this research, these mechanisms are expanded with a probability-driven adaptive operator selection, in which the choice between encircling, bubble-net assaulting, and random exploration is dynamically modified by their previous performance. Therefore, it indicates that exploration and exploitation are balanced throughout the iterative process. Table 2 displays the hyperparameter table for WOA.

Table 2: The hyperparameter table

Method	Hyperparameter	Notation / Variable	Description
WOA	Population size	*n*	Number of whales (search agents).
	Iterations	*d*	Maximum number of generations.

	Convergence factor	$\vec{\mathbf{E}}$	Linearly decreases from 2 to 0, balancing exploration/exploitation.
	Swing factor	$\vec{\mathbf{V}}$	Controls the movement step size in the position update.
	Random coefficient	$\vec{k}$	Random value in [0,1] for search randomness.
	Spiral constant	*(T')*	Determines spiral-shaped bubble-net updating.
	Probability threshold	*b*	Decides operator selection (spiral vs. random/encircling).

Predicting using long short-term memory (LSTM): LSTM is commonly used to handle gradient vanishing and explosion during complicated Prediction training. It needs global processing, and it is time-dependent. It has memory (hidden) cells that dynamically adjust the next step based on data from the previous stage. In the research, LSTM networks are used to forecast cost overruns, schedule deviations, and structural performance trends as shown in Figure 4. It consists of three gates: the input gate, the output gate, and the forget gate. The input gate is calculated by using Equation (2). LSTM is a recurrent neural network (NN) variation that performs well with sequential and time-dependent data. It addresses vanishing gradients using memory cells and three gates: input, forget, and output. In that research, LSTM is utilized to predict and produce candidate TS sequences while capturing dependencies between TC. An attention mechanism is included to highlight crucial test sequences, hence improving prediction accuracy.

$$i_t = \sigma(W_i[g_{t-1}, y_t] + c_i) \quad (2)$$

Where y_t signifies input data and g_{t-1} denotes the result of the preceding step. The input gate's bias and weight are denoted as c_i and W_i, respectively, whereas σ is a sigmoid function. i_t: Input gate vector (Vec). t: Time step. The data is further processed using the forget gate (f_t), as stated in Equation (3).

$$f_t = \sigma\left(W_f[g_{t-1}, y_t] + c_f\right) \quad (3)$$

W_f is the forget gate's weight, g_{t-1} is the preceding step's output, and the gate bias isc_f. f: forget gate. t: Time step.The following step states, as illustrated in Equation (4).

$$\bar{E}_t = tanh(W_E[g_{t-1}, y_t] + c_E) \quad (4)$$

Where $\bar{E}_t$ specifies the preceding step's state, c_E represents its bias, and W_E defines its weight. The previous state is used to determine the cell state at the present moment. $tanh$: Hyperbolic tangent function. Therefore f_t and $\bar{E}_t$ can be retrieved from Equations (1), (2), and (3), and they can be used in Equation (5) to calculate the current cell state.

$$E_t = f_t * E_{t-1} + i_t * \bar{E}_t \quad (5)$$

Lastly, the cell state is sent to the output gate as shown in Equation (6)

$$o_t = \sigma(W_o[g_{t-1}, y_t] + c_o) \quad (6)$$

Where o_t stands for the output gate, and W_o and c_o are its weight and bias, as shown in Equation (7), respectively.

$$h_t = o_t * tanh(E_t) \quad (7)$$

h_t: hidden state. o_t: Output gate. E_t: Cell state at time step t. Figure 4 presents the internal construction of an LSTM cell, with a focus on its three basic gates: forget, input, and output. The forget gate selects past information (C_{t-1}). To discard. The input gate determines which new information to store in the current cell state (C_t), with sigmoid and $tanh$ activations. The output gate determines which parts of the cell state contribute to the output (h_t). Together, these gates regulate memory flow over time steps.

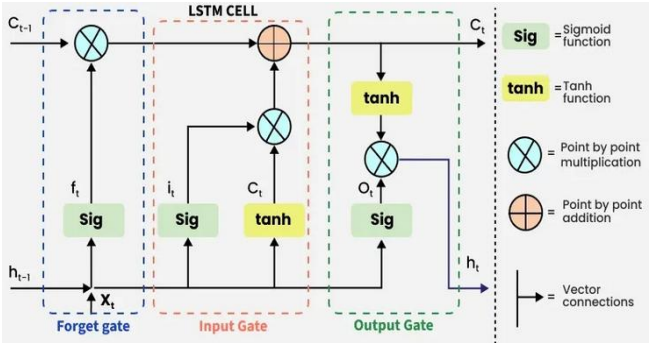

Figure 4: Architecture of LSTM

LSTM networks have memory cells that can remember things for long durations. Memory cells can inhibit or allow information storage through 3 gates: the input gate, the forget gate, and the output gate. LSTMs are effective when working with sequential data because this task typically requires learning how sequences evolve;

using an LSTM model to predict trends such as cost overruns, schedule deviations, and structural performance of civil engineering projects in Figure 4.

LSTM is used in the proposed framework to model sequential dependencies among the TC to help generate optimized TS. The critic network performs as an evaluator that measures the quality of randomly generated sequences, and it also helps instruct the LSTM to improve the prediction for accuracy and coverage. The constraints upon which the rules of TS are based are incorporated into the fitness function to enforce compliance with interaction requirements during selection. The incorporation of the critic network ensures that for each single generated TC, it adds as much interaction coverage as possible while keeping computational costs down and avoiding duplicate TC.

Attention mechanism: The process of attention is implemented by presenting a usual of attention weights that show the relative importance of each input component to the model's output. Before the input data travels complete the time out of the network, it is weighted using a set of coefficients. There are several approaches to include attention processes into a NN. A common method is to utilize a separate "attention layer" that accepts the input data and generates the attention weights based on a relevance or importance measure. The input data is then subjected to these weights before being sent across the remainder of the network. In this procedure, the dot product between each key Vec in the input data and the query Vec is evaluated as follows equation (8). To improve interpretability and prioritize essential aspects in sequential test data, an attention method is used. That applies adaptive weights to inputs, allowing the LSTM to prioritize the most informative signals during prediction and optimization.

$$Attention\ (R, L, U) = softmax\left(\frac{RL^S}{\sqrt{C_l}}\right)U \tag{8}$$

L Denotes the collection of key Vec is, U denotes the query Vec, and R normalizes the dot product between each key Vec and the query Vec. $softma$: The SoftMax function converts the raw score into a probability distribution. S: Similarity score. C_l: Scaling factor. The set of weights created by running the resulting dot products through a softmax function is then used to weight the value Vec s. The weighted sum of the value Vec is the attention mechanism's output (refer to Algorithm 1). Table 3 illustrates the hyperparameter table for LSTM.

Table 3: The hyperparameters for LSTM

Method	**Hyperparameter**	**Notation / Variable**	**Description**
LSTM	Input data	*y_t*	Sequence input at time *t*.
	Previous hidden state	*g_(t−1)*	Output of the preceding step.
	Input gate weight & bias	*W_i, c_i*	Parameters controlling input flow.
	Forget gate weight & bias.	*W_f, c_f*	Parameters controlling forgetting of past information.
	Candidate state weight & bias	*W_E, c_E*	Parameters for new state computation .
	Output gate weight & bias	*W_o, c_o*	Parameters controlling the final hidden output.
	Hidden state	*h_t*	LSTM output at time *t*.
	Cell state	*E_t*	Memory is retained across time steps.

The integration of LSTM and WOA combines sequential learning with adaptive search for effective TS optimization. LSTM captures temporal dependencies and generates candidate test sequences, while WOA evaluates and refines these sequences through encircling, spiral, and random operators. A probability table dynamically adjusts operator selection based on performance, ensuring balanced exploration and exploitation. This synergy enables efficient optimization, reduced redundancy, and improved fault detection, achieving highly accurate and reliable TS for complex Sof systems.

Algorithm

1: LSTM

Input: *The network receives previously processed training data.*
Output: *TS outcomes*
Step 1: *Input trained data and choose batch size, iteration intervals, hidden layer cell counts, and network layer counts.*

Step 2: *To alter the network's parameters, use the cross* −
entropy between the neural system's current output and the future input.

Step 3: *Continue with step* 2 *until the loss has converged and the loop is complete.*

Step 4: *Set the output weight parameters*

of the LSTM generator.t.

Feed previously processed sequences to the LSTM. Configure the batch size, epochs/iteration interval, hidden units, and layer depth. Forward-propagate to generate outputs. Calculate the cross-entropy of anticipated outputs and target labels (or next input

s). Back propagate across time to update parameters using an optimizer. Repeat the forward and backward passes until the loss convergence happens or patience runs out. Finally, apply trained generating weights to create efficient final test-suite sequences.

Whale optimization –long short-term memory (WO-LSTM): The unique feature of WO-LSTM is how it seamlessly incorporates a probability network into the TS network, allowing it to take full use of the LSTM method's dominance in the field of progression detection and the network's advantages in dealing with the constraints imposed by TS theory. The actor-critic network with the LSTM network as a generator is the first representation we provide, which results from the architecture and parameters of the TC creation model. The sequence prediction method is then built using LSTM hidden layers inference in line with the supplied weight matrix. We primarily concentrate on the control network, which introduces TC rules as constraints on the emergence of a particular style, computes the loss function, and develops a score method for a TS composed using networks. Because it consists of a WO-LSTM discriminator network and probability output, the probability network tuple can return when the created sequence comprises actual data. Figure 5 represents the proposed model. To address WOA's limitations of premature convergence and poor flexibility, use LSTM's sequential learning strength for directing just-in-time operator selection. The result is improved accuracy, recall, and robustness in restricted combinatorial test generation. The central purpose of integrating WOA with LSTM is to take advantage of WOA's capability for global exploration and LSTM's ability for sequential learning. This is achieved by:

- Using a probability chart that allows for adaptive operator selection (random, spiral, shrinker) based on previous performance, ensuring "just-in-time" adaptation.
- WOA as a discriminator, discerning the quality of the generated TS.
- LSTM as a generator, creating sequences based on learned sequence patterns;

Decrier net is reward-driven, which is designed to improve the proposed solutions and adhere to theoretical TS constraints. By integrating these approaches, we provide a robust general optimization framework that achieves high accuracy and recall while overcoming the classic limitations of WOA for premature convergence and weak early exploration.

Figure 5 shows the optimized TS using the WOA-LSTM. The process starts with the T-tuple table before progressing to TS generation and analysis. The TS selection phase uses the previous selection to select TC. The cases are then extracted and passed to WOA-LSTM, which uses advanced search methodologies. The output is a set of optimized TS while efficiently covering the interactions and minimizing redundancy and cost.

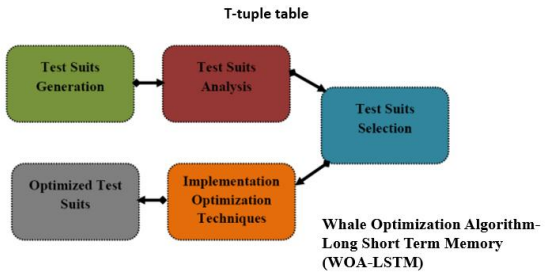

Figure 5: The T-tuple table analysis for the proposed method

In this model, the WO serves as a discriminator, and the LSTM network serves as a generator. While the WO-LSTM network struggles to evaluate a single or a series of TC sequences as a probability discriminator, LSTM generates TS sequences based on a certain probability distribution. Therefore, circular update mode and one-step updates are designed in this study. That is to say, the probability network and the critic network with TS rules serve as a reward system and take part in the modification of the created network to generate a complete TS sequence. By including the theoretical TS principles as the reward function and return scores, the critic network adjusts the weight parameters for generating sequences in TC.

The purpose of the WOA was to create a heuristic algorithm model that would mimic the predation behavior of humpback whales. The program mimics the spiral bubble-net foraging and hunting behaviors of humpback whales, including random, bubble, and encircling foraging. The WOA also uses the three search techniques mentioned above. The likelihood of each individual selecting one of the three behaviors is equal in the Whale optimization, where each decision variable represents the coordinates of each humpback whale. 2^4

The algorithm uses the leading whale position as the ideal outcome for the current group, while the remaining individuals utilize the leading whale position as the prey position. They finally encircle the prey by shifting their location and moving closer to it. The following is the behavior in equations (9-10)

$$\vec{Y}(d+1) = -\vec{T}.\vec{E} + \vec{Y} * (d) \tag{9}$$

$$\vec{T} = \left|\vec{V}.\vec{Y} * (d) - \vec{Y}(d)\right| \tag{10}$$

Where d represents the number of iterations, $\vec{Y}(d+1)$ is an individual after the $(d+1)$- position generation update; $\vec{Y}(d)$ is the d-generation's position as the leader whale. $\overrightarrow{E}$ is the convergence factor; $\vec{Y}(d)$ is the position of the person following the updating of the d-generation; Y represents the whale's position (solution). d: Iteration index. T: Distance between whale and prey. E: Convergence factor. V: Swing coefficient. $\vec{V}$: control randomness test. Directs candidate solutions to the best-known test configuration, providing efficient convergence in combinatorial optimization. $\vec{V}$ is the mathematical variable in equations (11-12), and is the swing factor $\overrightarrow{E}$ and $\overrightarrow{V}$.

$$\vec{V} = 2\vec{k} \tag{11}$$

$$\vec{E} = 1\vec{e} + 2\vec{k}.\vec{e} \tag{12}$$

Where $\overrightarrow{k}$ is an arbitrary number between [0, 1]; $\vec{e}$ linearly declines from 2 to 0 and gradually climbs during the iterative procedure. The spiral update formula takes whale predation by bubbles into account. The whale spirals upstream, changes positions, shoots forth bubbles of varying sizes, and contracts its cage to enclose its prey. The following is a formula for the location update in equations (13-14)

$$\vec{Y}(d+1) = \vec{Y} * (d) + \cos(2\pi f)\,.\,a^{pf}.\overrightarrow{T'} \tag{13}$$

$$\overrightarrow{T'} = \left|\vec{Y} * (d) - \vec{Y}(d)\right| \tag{14}$$

Y Indicates the current solution position Vec. $\overrightarrow{T'}$ Distance to optimal solution. f: Controlling the frequency of spiral

Oscillations. a The pace at which the spiral radius shrinks. p : The stochastic exponent scaling factor. d: The

Distance between a particular whale and the ideal individual, which is solved iteratively, and (T') is a spiral constant.

$\cos(2\pi f)$: Oscillatory term that modulates the spiral direction. Whales randomly adjust their locations to those of

Other whales when hunting, broadening their search area to find better prey. Pates whale positions through spiral

Motion around the best solution, balancing exploitation and exploration for robust combinatorial TS optimization. As

The convergence factor changes during the Whale optimization, each whale decides whether the search area has to be

Increased. When $|\vec{E}| > 1$, each whale will search the whole ocean to find its next meal. Here are the mathematical

Equations (15-16) for predatory randomness:

$$\vec{Y}(d+1) = -\vec{T}.\vec{E} + \vec{Y}_{rand}(d) \tag{15}$$

$$\vec{T} = \left|\vec{V}.\vec{Y}_{rand}(d) - \vec{Y}(d)\right| \tag{16}$$

$\vec{T}$: The distance between the randomly chosen whale and the whales whose solutions were found in the iterative procedure. $\vec{Y}_{rand}$: Whale chosen at random from the current population. In conclusion, the all-encompassing whale hunting approach in equation (17).

$$\vec{Y}(d+1) = \begin{cases} -\vec{T}.\vec{E} + \vec{Y} * (d) & , b > 0.5 \\ -\vec{T}.\vec{E} + \vec{Y}_{rand}(d) & , b < 0.5 \cup E < 1 \\ \vec{Y} * (d) + \cos(2\pi f) \,.\, a^{pf}.\overrightarrow{T'} & , b < 0.5 \cup E \geq 1 \end{cases} \quad (17)$$

$\vec{Y}(d+1)$: Indicates the whale's current position (solution) at iteration d. $\vec{Y} * (d)$: The best answer discovered thus far. $Y\vec{Y}_{rand}(d)$: randomly chosen whale position. $\overrightarrow{T'}$: Distance Vec (to prey/best solution). $\vec{E}$: Convergence factor (balances exploration and exploitation). b: The probability parameter in [0, 1] governs the operator selection.f Represents the spiral oscillation parameter. Spiral shrinkage and stochastic control factor are denoted by a and p, respectively. This unified equation adaptively chooses encircling, random search, or spiral strategies to ensure balanced exploration-exploitation for effective TS optimization. In addition, the discriminator calculates the likelihood that a synthetic sequence represents the real thing and reports that likelihood to the user. Algorithm 2 denotes the pseudo-code of the proposed model.

The WOA was inspired by humpback whales' cooperative foraging technique, specifically, their unique bubble-net hunting behavior. That depicts three essential behaviors—surrounding prey, spiral bubble-net attack, and random exploration—that balance exploitation and exploration in the search space. 1. Encircling Prey (Exploitation Phase). Whales can detect the presence of prey and circle it. WOA accepts that the prey position is the best current solution. Other alternative solutions change their positions relative to the optimal solution in equation (18)

$$y(d+1) = y^*(d) - E \cdot T \quad (18)$$

$y^*(d)$: The optimal solution at iteration d $.Y(d)$ represents the current whale's position. $|T = | V \cdot Y * (d) - Y(d)|$: The space among the whale and the prey E is the convergence factor that regulates exploitation. V: The swing coefficient that introduces variability. d: Iteration index. The process directs whales to the optimal global answer, highlighting the need for increased search intensity. Moves candidate solutions closer to the optimal solution, increasing exploitation to accelerate convergence in combinatorial TS optimization.

2. Spiral Bubble-Net Aggressive (Exploitation with Diversity): Humpback whales use a spiral-shaped bubble net to capture fish. WOA mimics this by creating a logarithmic spiral trajectory around the prey in equation (19)

$$y(d+1) = y^*(d) + \cos(2\pi f).\, a^{pf}.\, T' \quad (19)$$

$T' = | Y * (d) - Y(d) |$: The distance from prey a and pf pf regulates spiral shape and diminishing movement. f introduces oscillations to enhance diversity. $Cos\ (2\pi f)$ models the oscillatory spiral motion around the prey. f: Random number in [0, 1] controlling the shape of the spiral (frequency/angle of movement). The modified rule allows candidate solutions to circle the optimal solution with a decreasing radius, resulting in convergence without premature stagnation. Models adaptive spiral motion around optimal solutions, combining convergence and diversity to avoid stagnation in the solution search space.

3. Random Search (Exploration Phase): To prevent local optima entrapment, whales may search randomly for prey. This is modeled in equation (20).

$$y(d+1) = y_{rand}(d) - E.T \quad (20)$$

$y(d+1)$ Updates the whale's location for the next iteration. $y_{rand}(d)$: Indicates the location of a randomly selected whale from the population at iteration d. E is the convergence factor that controls the movement step size, which decreases during iterations $|y(d+1) = y_{rand}(d)|$: The distance between the current whale and the randomly chosen whale, with V being the swing factor that scales variability. Encourages exploration by guiding solutions toward random peers, enhancing search diversity and avoiding previous union in the optimization process.

Stability amongst Exploration and Exploitation: The probability parameter and adaptive convergence factor (E) determine the transition between these mechanisms, which decreases with each repetition. Early stages prioritize exploration (random searches and spiral motions), but later stages prioritize exploitation (encircling the best answer). Together, these features make WOA a strong global optimization algorithm capable of avoiding premature conjunction and efficiently traversing high-dimensional search spaces. The hybrid approach of the WOA-LSTM makes use of metaheuristic search and adaptive learning to avoid challenges related to premature convergence, exploration limitations, and operator adaptation ineffectiveness. WOA offers the context of a population-based search, while LSTM offers dynamic operator selection by maintaining an operator probability table successfully based on the performance of that operator in the past. Z-score normalization ensures the features are scaled for stability and consistency, as well as allowing the training to be effective in an ultimately balanced form of data representation. The contribution of the method would enhance exploration-exploitation balance algorithms' ability to prevent being stuck in the early iterations that take place in search, while also delivering the best optimization accuracy, meaning the WOA-LSTM algorithm provided empirically competitive results

previously perceived as a highly complex combinatorial test generation task.

3.6 Statistical tests using the t-test

The t-test is presented to statistically analyze the importance of differences between the proposed WOA-LSTM method and baseline techniques such as traditional WOA and other metaheuristic techniques in terms of accuracy, recall, and fault detection rate. Dependent-samples t-tests and independent-samples t-tests can also be used according to the type of evaluation: independent means when comparing outcomes from different datasets or through an algorithm; and repeated measures when evaluating outcomes from the same dataset. The t-test calculates the t-value from mean differences, standard deviation, and sample size. The statistic value then corresponds to a p-value. The appropriate approach would be to declare statistical significance as a p-value less than or equal to a significance level ($\alpha = 0.05$). This means when the p-value is less than 0.05, there is a statistically significant improvement in the results. Doing this gives reason to be confident in validating that the proposed WOA-LSTM method's improvements in TS generation, optimization efficiency, and predictive accuracy are statistically significant and robust evidence that the enhancements to WOA-LSTM performance are performance gains in terms of improved TC generation and are not due to random sampling variances. There is also value in using effect size measures such as Cohen's d in conjunction with the t-test to quantify the magnitude of the improvement between methods to assess both significance and practical value.

Algorithm 2: Pseudo-code of the WOA-LSTM algorithm

$Initialize\ the\ whales\ population\ X;\ (i = 1, 2, \ldots, n)$
$Calculate\ the\ fitness\ of\ each\ search\ agent.$
$X* =\ the\ best\ search\ agent.$
$Xcurrent\ =\ the\ current\ search\ agent.$
$episode\ =\ 0.$
$for\ each\ state\ S = [s1, s2, .., sn]\ and\ actions\ A\ =\ [a1, a2, \ldots, an]\ do$
$set\ CA(N, 2n)\ =\ 0$
end
$while\ i\ <\ maximum\ number\ of\ iterations\ do$
$if\ (episode\ <\ 1)\ then$
$Select\ an\ action\ and\ a\ state\ randomly$
$else$
From the current state s, select the best action a from t-table if (action == shrinking mechanism) then
$Update\ the\ current\ search\ agent\ using$
$else\ if\ (action\ ==\ spiral - shaped\ mechnism)\ then$
$Update\ the\ current\ search\ agent\ using$
$else\ if\ (action\ random\ generation)\ then$
$Select\ a\ random\ search\ agent\ ()$
$Update\ the\ current\ search\ agent\ using$
$Check\ if\ search\ agent\ goes\ beyond\ the\ search\ spa$
$Calculate\ the\ fitness\ of\ the\ current\ search\ agent$
$if\ (Xcurrent\ better\ than\ X)\ then$
$人 = 1$
$X* =\ Xcurrent$
$non - improvement\ =\ 0$
$else$
$Generate\ a\ random\ number\ randNum\ in\ [0\ 1]$
$if\ (randNum\ \leq\ e - {}^{\delta})\ then$
$X* =\ Xcurrent$
$人 = 1$
$non - improvement\ =\ 0$
$Else$
${}^{\delta} =\ Xcurrent - X*$
$non - improvement\ ++$
$if\ (non - improvement\ ==\ Max - non - improvement)\ then$
$人++$
$non - improvement\ ++$
$Get\ immediate\ r\ using$
$Get\ the\ maximum\ t\ value\ for\ the\ next\ state\ s + 1$
$Update\ a\ using$
$Update\ t - table\ entry\ using$
$t = t + 1$
$episode\ ++$
$return\ X*$

The method begins by initializing the whale population and assessing the fitness of each agent. The best whale is saved as the global best (X^*), while the current whale is set as $X_{current}$. States and actions are established, and credit/action-value tables are set up. If episode =0, actions are picked at random; otherwise, the optimal action is chosen from the t-table. $X_{current}$ is updated using shrinkage (Eq.2), spiral (Eq.6), or a random mechanism (Eq.9), depending on the action taken. Boundary inspections and fitness assessments follow. If $X_{current}$ improves, it updatesX^*; otherwise, acceptance is done probabilistically with $e - {}^{\delta}$. Non-improvement numbers drive diversification. Rewards are computed (Eq. 14), and the t-table is updated using temporal-difference learning. Episodes continue until convergence, resulting in the ideal (X^*).

4 Results and discussion

The effectiveness of WOA-LSTM is evaluated in this part in comparison to other well-known population-based meta-heuristic procedures and untainted computational strategies. The size of the CA (i.e., TS size) was used to

gauge efficiency. Two additional metrics have been supplied: (1) the suggested WOA-LSTM strategy's convergence behaviour, and (2) the strategy's suitability for the creation of combinatorial test data.

Experiments are broken down into four sections. First, methodically adjusted the EMC algorithm's max-non-improvement value for the acceptance probability. Second, assessed and contrasted the WOA-LSTM approach with the population-based meta-heuristic algorithms that are already in use. Third, compare the proposed approach to other constraint-supporting techniques that are currently in use. Finally, assessed the WOA-LSTM strategy's suitability and efficacy for combinatorial test data creation.

4.1 Experimental setup

The WOA-LSTM was showed on a system with an Intel Core i7 processor, 16 GB RAM, and 1 TB SSD storage to ensure efficient data handling and execution. Python 3.10 was used with TensorFlow and Scikit-learn libraries. Data preprocessing included Z-score normalization for feature scaling and categorical encoding for input preparation. The experiments leveraged GPU acceleration when available, ensuring faster training and evaluation of the hybrid optimization-learning framework.

4.2 Parameter tuning

Most meta-heuristic algorithms continue to require population size (PS) and a extreme number of iterations (Ite). Based on previous research, the concentrated number of Ite and PS to 100 and 180, respectively. Another parameter to examine is the maximum non-improvement parameter in the LSTM algorithm's receiving probability. This value represents the number of Ite in which the answers were not improved. A minor max-non-improvement (MNI) could prevent the best candidate solution from being found, whilst a large one could trap the candidate solution in local optima. Therefore, when picking MNI, careful coordination is required. Table 4 demonstrates the parameter configurations for various metaheuristic algorithms used in TS optimization, detailing (PS) and Ite count for DPSO, CS, ABCVS, APSO, and QLSCA methods.

Table 4: Extracted from algorithmic performance tuning studies in optimization research.

Algorithm	**Parameters**	**Values**
DPSO	PS	80
	Ite	250
CS	PS	100
	Ite	100
ABCVS	PS	50
	Ite	80
APSO	PS	80
	Ite	100
QLSCA	PS	40
	ite	100

Best and average TS size:

The WOA-LSTM approach was run 30 times with different maximum non-improvement values (i.e., 5, 10, 20, 30, 40, 50, and 60) tested. Table 5 shows the best and average TS sizes, with darkened cells representing the best size. The implementation time is stated in seconds, and Table 3 includes the best and average execution times. CA (N; 2, 46) yielded the greatest results with MNI values of 20 and 30, whereas CA (N; 2, 57) yielded the best results with MNI of 30. Considering the duration, the optimal MNI was decided, which is thirty. Performance comparison of TS generation for CA (N; 2, 46) and CA (N; 2, 57), displaying the best and average sizes and execution durations for various stagnation criteria.

Table 5: Based on experimental evaluations of combinatorial TS creation.

	CA(N; 2, 4^6)				**CA(N; 2, 5^7)**			
	Best		**Average**		**Best**		**Average**	
MNI	**Size**	**Time**	**Size**	**Time**	**Size**	**Time**	**Size**	**Time**
5	24	37.2	25.7	38.9	41	113.6	43	122.46
10	24	37.9	25.7	40.1	39	111.3	41	118.1
20	24	38.5	25.8	38.1	37	113.6	40	121.8
30	24	36.6	25.6	38.2	37	111.8	39	119.9
40	25	38.2	25.9	41.8	38	109.7	41	118.3
50	25	38.3	26.1	41.7	39	117.7	41	122.8
60	25	39.1	25.9	41.4	41	111.1	43	119.9

- Performance analysis

To verify the efficacy of the proposed strategy, WOA-LSTM, a comparison analysis is performed against other well-known AI techniques. The techniques include Artificial Bee Colony Based Variable Strength (ABCVS) [28], Discrete Particle Swarm Optimization (DPSO) [27], Adaptive Particle Swarm Optimization (APSO) [30], and

Q-Learning Sine Cosine Algorithm (QLSCA) [29],. The test results of the dataset techniques on image and language recognition in the evolution of AI to recover the datasets using the aforementioned classifiers are summarized accordingly. Table 6 lists the criteria used to evaluate the proposed model, including accuracy, F1-score, precision, and recall precision. One of the most important performance measures for classification is accuracy.

Table 6: Results from an experimental investigation of optimization algorithm performance.

Methods	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**	**R^2**
DPSO	87.7	86.3	87.3	86.8	96.35
ABCVS	92.4	91.4	92.2	92.1	86
QLSCA	94.7	93.5	94.4	94.6	98.7
APSO	93.8	92.6	95.6	97.6	96.7
WOA-LSTM [Proposed]	99.3	98.7	99.1	98.8	98.9

Figure 6 shows how the suggested WOA-LSTM model batter than existing methods across a variety of calculation metrics. With an accuracy of 99.3%, the suggested version outperforms the existing method, indicating its sophisticated ability to efficiently categorize instances. Furthermore, the WOA-LSTM has the highest precision (98.7%), and recall (99.1%) implying that it may be capable of detecting relevant times while minimizing false positives and false negatives. The WOA-LSTM model achieves a much higher F1 score (98.8%), indicating an improved balance of precision and F1 score. After associated to existing methodologies, the Suggested model outperforms them all performances metrics indicating its classification usefulness.

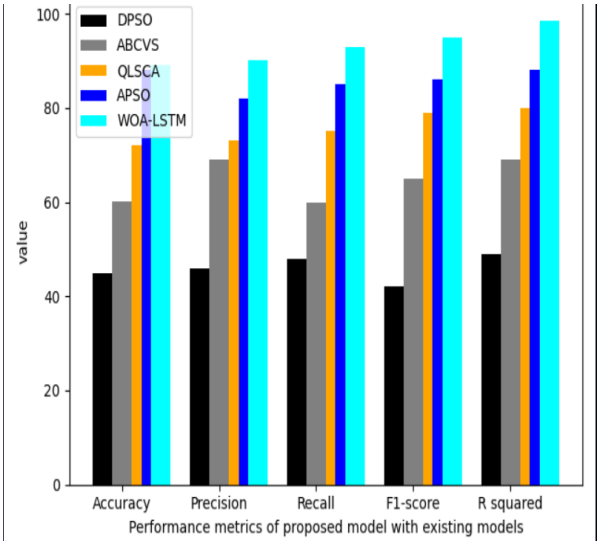

Figure 6: Metaheuristic techniques' performance is compared for TS optimization.

T-test: Table 7 shows the detailed statistical research comparing the performance of the proposed WOA-LSTM model to different existing meta-heuristic algorithms using a t-test. Performance measures such as accuracy, precision, recall, F1-score, and R^2 are obtainable as mean ± standard deviation across 30 independent runs. WOA-LSTM produced exceptional performance values, including 99.3% accuracy, 98.7% precision, 99.1% recall, and 98.8% F1-score. The table also contains t-values, degrees of freedom, p-values, implication levels, and effect sizes (Cohen's d), which show that WOA-LSTM outperforms previous models in TS optimization, resilience, and fault detection.

Table 7: Statistical analysis comparing WOA-LSTM performance to established optimization techniques.

Metric	**Existing Method**	**Existing Mean ± SD**	**WOA-LSTM Mean ± SD**	**t-value**	**df**	**p-value**	**Effect Size (Cohen's d)**
Accuracy (%)	DPSO [27]	87.7 ± 1.2	99.3 ± 0.35	15.62	29	<0.001	4.15
Precision (%)	ABCVS [28]	91.4 ± 1.0	98.7 ± 0.33	12.03	29	<0.001	3.6
Recall (%)	QLSCA [29]	94.4 ± 0.9	99.1 ± 0.40	10.25	29	<0.001	3.25
F1-score (%)	APSO [30]	97.6 ± 0.95	98.8 ± 0.38	4.21	29	0.0003	1.1
R^2	QLSCA [29]	96.35 ± 1.2	98.9 ± 0.4	6.95	29	<0.001	1.9

➢ **Error metrics**

- The value of the coefficient of determination (R^2) is an indicator of statistical significance commonly used to assess perfect fit. It displays a proportion of the variance of a reliant on parameter that could be described by a few autonomous variables in a NN model. The proposed WOA-LSTM method has a R^2 value of 98.7, showing high explanatory power for the dependent variable. Existing models, such as QLSCA and APSO, have R^2 values of 96.35 and 86, respectively.
- The Mean Squared Error (MSE) parameter helps optimize TS operations by determining how much of a dataset's total variance a model explains. That provides information about the efficiency and reliability of types for picture and language recognition. The WOA-LSTM simulation's stated MSE score of 0.05 demonstrates its efficacy in describing the dataset's overall variation. Existing models, such as QLSCA (0.28 MSE) and APSO (0.17), perform poorly in terms of variance capture. Table 8 and Figure 7 show the MSE findings.

Table 8: The MSE value of the proposed method.

METHODS	***MSE***
DPSO	0.51
ABCVS	0.39
QLSCA	0.28
APSO	0.17
WOA-LSTM [Proposed]	0.05

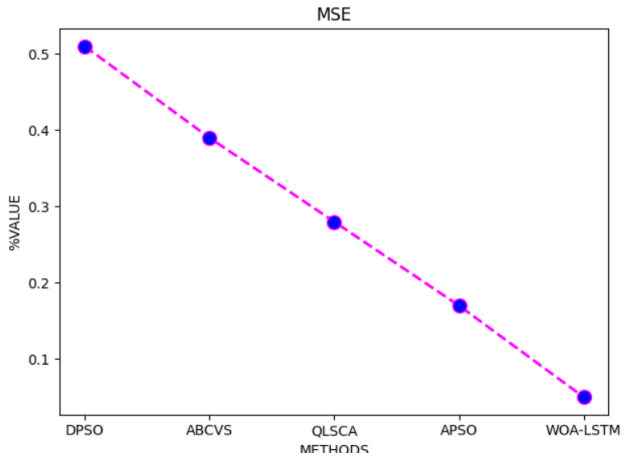

Figure 7: Graphical outcomes of MSE

- *The Mean Absolute Error (MAE) is a simple method for evaluating the range of AI-based voice recognition errors by measuring the regular absolute differences between observed and anticipated results. Because MAE changes all faults similarly, it is less prone to outliers than RMSE, which could prove advantageous in some situations. The recommended MAE model provides a value of 0.01 to demonstrate its excellence in catching real values. WOA-LSTM has a greater DPSO of 0.0271 than APSO of 0.13, demonstrating that the proposed technique performs higher in minimizing absolute errors.*
- *Root Mean Square Error (RMSE) is a statistic used to compare forecasts to actual results. It determines the average size of errors among predictable and practical values. To emphasize larger errors caused by forming, it squares the difference between the expected and real numbers, averages them, and then estimates the square root. The proposed model has an RMSE of 0.013, indicating a relatively low prediction error. In comparison, QLSCA has a higher RMSE of 0.17 than APSO of 0.127, demonstrating that the proposed method is active at decreasing development errors. Table 9 and Figure 8 display the results of RMSE and MAE.*

Table 9: Numerical outcomes of RMSE and MAE

METHODS	RMSE	MAE
DPSO	0.03	0.02
ABCVS	0.15	0.11
QLSCA	0.17	0.09
APSO	0.12	0.13
WOA-LSTM [Proposed]	0.013	0.01

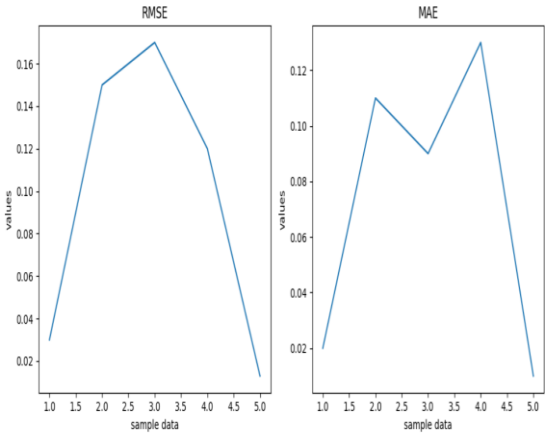

Figure 8: Graphical outcomes of RMSE and MAE

Parameter explanation

Size of Test Suite: Minimizing TS size enables thorough coverage with fewer cases, which is consistent with efficient combinatorial testing by decreasing redundancy while preserving accuracy and fault detection capacity. Execution cost: lower execution costs reflect fewer computational resources and faster optimization, which helps us achieve creating an adaptive, cost-effective WOA-LSTM model for scalable, real-world software testing applications and optimization activities.

Table 10 and Figure 9 display the performance evaluation of the proposed WOA-LSTM vs existing methods include genetic algorithm for web service composition (G_WSC), genetic algorithm (G), hybrid genetic swarm or hybrid genetic strategy (HGS), greedy algorithm (GRE), and particle swarm optimization (PSO) to evaluate the size of t TS. The proposed WOA-LSTM utilizes the smallest TS size (5.0), providing the maximum reduction performance.

Table 10: The size of TS values for proposed vs existing methods.

Methods	Size of Test Suite
G_WSC [16]	6.0
G [16]	5.6
HGS [16]	5.5
GRE [16]	5.4
PSO [16]	5.3
WOA-LSTM [Proposed]	5.0

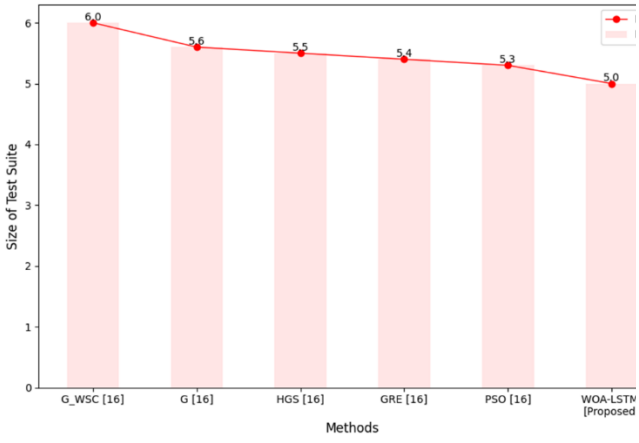

Figure 9: The size of TS values for the proposed method.

Table 11 and Figure 10 illustrate the execution cost contrast. The WOA-LSTM algorithm achieved the lowest execution cost of 45, indicating the most efficient performance of all existing algorithms.

Table 11: The execution cost values for the proposed method.

Methods	Execution Cost
G_WSC [16]	70
G [16]	60
HGS [16]	55
GRE [16]	75
PSO [16]	50
WOA-LSTM [Proposed]	45

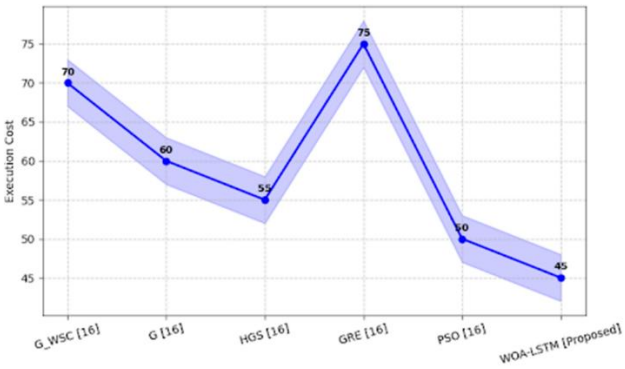

Figure 10: The execution Cost values for the proposed method.

Advantages of using WOA for TS Optimization:

- Effective for global optimisation: Even in challenging TS optimisation, WOA's global search feature helps in locating nearly ideal solutions.
- Simple and easy to implement: WOA is easier to modify for various Sof testing scenarios because it is less variable-dependent and has a direct implementation.
- Effective for fault detection: WOA can assist in choosing TC that are more likely to reveal flaws, resulting in testing that is more effective.

Applications:

1. WOA can assist in lowering the quantity of TC required to cover a Sof system, hence saving resources and time. 2. WOA enables testers to concentrate on the most important cases by ranking TC according to their probability of finding errors. 3. After Sof modifications, WOA can be used to choose the most pertinent TC for regression testing, guaranteeing that new flaws are identified promptly and effectively. Challenges: The WOA is a good fit to serve as the foundation algorithm for our new TW strategy because of its strong worldwide search capabilities. WOA occasionally converges to local optima, which might not be the optimal solution, similar to other metaheuristics. WOA can be costly to compute, particularly for large TS. WOA settings have the potential to impact its performance, necessitating careful adjustment for certain issues. WOA is a strong and adaptable algorithm that offers advantages in efficiency, fault identification, and global search capabilities for Sof engineering TS optimisation. To further increase its performance in particular testing circumstances, it is crucial to take into account probable difficulties and investigate better variations.

4.3 Discussion

The results of the experiments clearly show that the WOA-LSTM outperforms previous population-based metaheuristic algorithms for combinatorial TS generation for program testing. Previous traditional methods, DPSO [27], ABCVS [28], QLSCA [29], and APSO [30], had inherent challenges of premature convergence, not exploring enough in the early search stages, and getting stuck in local minima. These challenges limited their ability to produce TS containing a minimal number of TC, while still being comprehensive. If reducing the number of TC compromise's fault detection effectiveness, then it makes little difference when generating a TS. The WOA-LSTM method's basic mechanism is that the WOA algorithm allows the LSTM-based probability to drive the decision on which search operation to select, making it able to explore and then exploit as needed. Dynamic adaptation can prevent stagnation and improve convergence behaviour, so the overall volume of TC was smaller while the overall efficiency of the generated tests improved. In addition, WOA-LSTM consistently outperformed the previous ACO and metaheuristic methods across the evaluation criteria; that is, classification accuracy, recall, and robustness metric results indicated a stronger ability in detecting the relevant patterns in TC with fewer false-positive or false-negative values. The inclusion of metaheuristic adaptive memory into the previous search-based dimension provided a more robust and flexible optimization option to the limitations of previous methods. The WOA-LSTM was able to correct the weaknesses of previous option-based methods to provide a framework, and hence system, for scalable and effective approaches to combinatorial test generation. Moreover, it illustrates the wider potential base usage of optimizing any Sof test generation advancement and related optimization problems.

5 Conclusion

The hybrid optimization framework using WOA-LSTM to produce optimized combinatorial TS developments with increased accuracy, reduced redundancy, and an adaptive operator selection proficiency. The existing's researches shown in DPSO [21], ABCVS [22], APSO [23], and QLSCA [24] were clearly successful in optimizing combinatorial TS, but all dealt with significant issues including stagnation due to premature convergence, poor adaptive exploration across many iterations, and the ability to model sequential dependencies during test generation. Our proposed WOA-LSTM provides a solution to these problems by leveraging WOA's global exploration ability and combining that with LSTM's sequential learning ability to produce a more robust and dynamic operator selection mechanism that provides a probabilistic basis and is better adapted than prior studies. WOA-LSTM can avoid stagnation in the first several iterations of the combinatorial TS, while maintaining voltage exploration and exploitation. Furthermore, the experimental results show that WOA-LSTM has better optimization overall with 99.3% accuracy, 99.1% recall, 98.75% as a precision measure, and 98.92% as an F1 score- when compared to the research such as DPSO [21], ABCVS [22], APSO [23], and QLSCA [24]. However, the likelihood remains that the cost of computational resources during the LSTM training task is a higher cost than the combinatorial testing, dependent on available computing power, RAM, and the centuries spent in hyperparameter tuning. The higher computational burden, hyper-parameter sensitivity, and possible scalability problems with very large datasets. Plus, the proposed WOA-LSTM relies on the historical performance of operators, which may have limitations on the ability to adapt to rapidly changing exploratory spaces.

Traditional metaheuristics like DPSO, ABCVS, QLSCA, and APSO often exhibit premature convergence, weak exploration, and little adaptability when generating optimal TC for complex Sof systems. This is particularly relevant in automated Sof testing, where efficient fault detection and variance in test run numbers are crucial. The proposed method, WOA-LSTM, offers a robust optimization method for finding test generations in complex Sof systems, balancing exploration and exploitation. This method is particularly important in safety-critical environments like healthcare,

transportation, and financial systems, ensuring reliable, efficient, and scalable test generation.

Future work will focus on improving this training time, calling on possible reinforcement learning based approaches to provide better adaptive parameter controls, and validating the WOA-LSTM framework on industrial-scale real-world applications.

References

[1] Chakraborty S, Saha AK, Chhabra A (2023) Improving whale optimization algorithm with elite strategy and its application to engineering-design and cloud task scheduling problems. Cogn Comput. https://doi.org/10.1007/s12559-023-10123-x

[2] Paul C, Roy PK, Mukherjee V (2023) Wind and solar based multi-objective hydro-thermal scheduling using chaotic-oppositional whale optimization algorithm. Electr Power Compon Syst 51(6):568–592. https://doi.org/10.1080/15325008.2023.2178901

[3] Li M, Yu X, Fu B, Wang X (2023) A modified whale optimization algorithm with multi-strategy mechanism for global optimization problems. Neural Comput Appl. https://doi.org/10.1007/s00521-023-08456-7

[4] Nadimi-Shahraki MH, Zamani H, Fatahi A, Mirjalili S (2023) MFO-SFR: an enhanced moth-flame optimization algorithm using effective stagnation finding and replacing strategy. Mathematics 11:862. https://doi.org/10.3390/math11040862

[5] Asghari Varzaneh Z, Hosseini S, Javidi MM (2023) A novel binary horse herd optimization algorithm for feature selection problem. Multimedia Tools Appl. https://doi.org/10.1007/s11042-023-16342-9

[6] Devi RM, Premkumar M, Kiruthiga G, Sowmya R (2023) IGJO: an improved golden jackel optimization algorithm using local escaping operator for feature selection problems. Neural Process Lett. https://doi.org/10.1007/s11063-023-11345-2

[7] Liu J, Shi J, Hao F, Dai M (2022) A novel enhanced global exploration whale optimization algorithm based on Lévy flights and judgment mechanism for global continuous optimization problems. Eng Comput. https://doi.org/10.1007/s00366-022-01645-9

[8] Nadimi-Shahraki MH, Zamani H, Varzaneh ZA, Mirjalili S (2023) A systematic review of the whale optimization algorithm: theoretical foundation, improvements, and hybridizations. Arch Comput Methods Eng 30:4113–4159. https://doi.org/10.1007/s11831-023-09992-5

[9] Huang W, Li J, Liu D (2023) Real-time solution of unsteady inverse heat conduction problem based on parameter-adaptive PID with improved whale optimization algorithm. Energies 16:225. https://doi.org/10.3390/en16010225

[10] Nadimi-Shahraki MH (2023) An effective hybridization of quantum-based avian navigation and bonobo optimizers to solve numerical and mechanical engineering problems. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2023.120598

[11] Liang ZY, Shu T, Ding ZH (2024) A novel improved whale optimization algorithm for global optimization and engineering applications. Mathematics 12:636. https://doi.org/10.3390/math12040636

[12] Qu SZ, Liu H, Xu YH, Wang L, Liu YF, Zhang LN, Song JF, Li ZS (2024) Application of spiral enhanced whale optimization algorithm in solving optimization problems. Sci Rep 14:24534. https://doi.org/10.1038/s41598-024-24534-2

[13] Mafarja M, Thaher T, Al-Betar MA, Too J, Awadallah MA, Abu Doush I, Turabieh H (2023) Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning. Appl Intell 53:18715–18757. https://doi.org/10.1007/s10489-023-05566-1

[14] Zhao F, Xu Z, Bao H, Xu T, Zhu N (2023) A cooperative whale optimization algorithm for energy-efficient scheduling of the distributed blocking flow-shop with sequence-dependent setup time. Comput Ind Eng 178:109082. https://doi.org/10.1016/j.cie.2023.109082

[15] Liu J, Shi J, Hao F, Dai M (2023) A novel enhanced global exploration whale optimization algorithm based on Lévy flights and judgment mechanism for global continuous optimization problems. Eng Comput 39:2433–2461. https://doi.org/10.1007/s00366-023-01789-1

[16] Deneke A, Assefa BG, Mohapatra SK (2022) Test suite minimization using particle swarm optimization. Mater Today Proc 60:229-233. https://doi.org/10.1016/j.matpr.2021.12.472

[17] Ahmed M, Nasser AB, Zamli KZ (2022) Construction of prioritized T-way test suite using bi-objective dragonfly algorithm. IEEE Access 10:71683-71698. https://doi.org/10.1109/ACCESS.2022.3188856

[18] Zayed HAB, Maashi MS (2021) Optimizing the software testing problem using search-based software engineering techniques. Intell Autom Soft Comput 29(1). http://dx.doi.org/10.32604/iasc.2021.017239

[19] Ouyang J, Lin H, Hong Y (2024) Whale optimization algorithm BP neural network with chaotic mapping improving for SOC estimation of LMFP battery. Energies 17:4300. https://doi.org/10.3390/en17124300

[20] Zamani H, Nadimi-Shahraki MH, Gandomi AH (2022) Starling murmuration optimizer: a novel bio-inspired algorithm for global and engineering optimization. Comput Methods Appl Mech Eng 392:114616. https://doi.org/10.1016/j.cma.2022.114616

[21] Abdollahzadeh B, Gharehchopogh FS, Khodadadi N, Mirjalili S (2022) Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. Adv Eng Softw 174:103282. https://doi.org/10.1016/j.advengsoft.2022.103282

[22] Gharehchopogh FS, Nadimi-Shahraki MH, Barshandeh S, Abdollahzadeh B et al (2022) CQFFA: A chaotic quasi-oppositional farmland fertility algorithm for solving engineering optimization problems. Comput Intell Neurosci 2022:451–464. https://doi.org/10.1155/2022/451464

[23] Cao D, Xu Y, Yang Z, Dong H et al (2022) An enhanced whale optimization algorithm with improved dynamic opposite learning and adaptive inertia weight strategy. Complex Intell Syst 2(3):142–156. https://doi.org/10.1007/s40747-022-00948-3

[24] Wang J, Bei J, Song H, Zhang H et al (2023) A whale optimization algorithm with combined mutation and removing similarity for global optimization and multilevel thresholding image segmentation. Appl Soft Comput 137:110130. https://doi.org/10.1016/j.asoc.2023.110130

[25] Liu D, Zhou S, Shen R, Luo X (2023) Color image edge detection method based on the improved whale optimization algorithm. IEEE Access 11:5981–5989. https://doi.org/10.1109/ACCESS.2023.3256781

[26] Burachynskyi A, Shantyr A (2025) Overview of artificial intelligence application methods in software development. Informatica 49(28). https://doi.org/10.31449/inf.v49i28.8694

[27] Wu H, Nie C, Kuo FC, Leung H, Colbourn CJ (2014) A discrete particle swarm optimization for covering array generation. IEEE Trans Evol Comput 19(4):575–591. https://doi.org/10.1109/TEVC.2014.2311752

[28] Alazzawi AK, Rais HM, Basri S (2019) ABCVS: an artificial bee colony for generating variable t-way test sets. Int J Adv Comput Sci Appl 10(3):123–132. https://doi.org/10.14569/IJACSA.2019.0100315

[29] Zamli KZ, Din F, Ahmed BS, Bures M (2018) A hybrid Q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem. PLoS ONE 13(5):e0195675. https://doi.org/10.1371/journal.pone.0195675

[30] Mahmoud T, Ahmed BS (2015) An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing. Exp Syst Appl 42(22):8753–8765. https://doi.org/10.1016/j.eswa.2015.07.011