Engineered Science
DOI: https://dx.doi.org/10.30919/es933

# Preserving Security in Internet-of-Things Healthcare System with Metaheuristic-Driven Intrusion Detection

Nidhi Goswami,[1] Sahil Raj,[2] Deepti Thakral,[3] José Luis Arias-Gonzáles,[4] Judith Flores-Albornoz,[5] EdwinAsnate-Salazar,[6] Dhiraj Kapila,[7] Sanjay Yadav[8] and Surendra Kumar[9,*]

## Abstract

The Internet of Things (IoT) makes IoT devices more vulnerable to cyberattacks, especially Distributed Denial of Service (DDoS), raising privacy and security issues. Application-layer DDoS: zombie machines submit queries to the affected server. IDS cannot detect these requests because TCP connections are valid. In this research, we propose the Lion-Salp-Swarm-Optimization Algorithm (LSSOA), which utilizes the freely available IoT-Flock software to create a dataset containing both legitimate and malicious traffic. We employ metaheuristic algorithms like Lion optimization, Whale optimization, Spider-Monkey optimization, and Salp Swarm optimization to detect online attacks and defend the Internet of Medical Things (IoMT) environment. Our framework accurately detects attacks while reducing false positives by overcoming intrusion detection restrictions. Our metaheuristic algorithm outperforms others. Our method is useful for Internet of Things-based enterprises. Overall, the LSSOA framework provides a powerful tool to detect and prevent cyber-attacks in the IoMT environment, demonstrating the potential benefits of novel intrusion detection techniques. Our study emphasizes the importance of enhancing IoT security, particularly in critical industries like healthcare, and highlights the need for continuous efforts to develop effective and innovative approaches to address emerging cyber threats.

*Keywords:* Internet of medical things; Distributed denial of service; Internet of things; Metaheuristic; Lion-Salp-Swarm-optimization algorithm.
Received: 05 May 2023; Revised: 02 August 2023; Accepted: 02 August 2023.
Article type: Research article.

## 1. Introduction

Base station communications, Wireless, and Wi-Fi networks are used to connect the network devices that make the framework for the World Wide Web.[1] The Internet of Things (IoT) gadgets are extensively used in gadgets (thermostats, fridge, etc.), security systems, healthcare, computer peripherals, the military, agriculture, and other fields.[2,3] The Internet Protocol (IP) is used by these IoT devices to convey information from one location to another. By identifying the machine, this IP protocol enables quick communication without the need for human interaction. Yet, IoT devices alter human life in a variety of applications, and IoT security risks are serious. The current crop of intrusion detection systems might not be sufficient to fend off the sophisticated threats of today. As a result, the risk of IoT infiltration has been evaluated in this study by using an inadequate convoluted web to defend against malicious attacks. Each IoT gadget has unique qualities, like large data collection, connectivity to both real-world and online world, creation of complicated environments, and centralized architecture.[4] These features make the IoT function effectively, but they also make communication between threat actors more vulnerable. According to the Systematic Literature Review (SLR) Blockchain study,[5] the global population uses almost a billion IoT devices to enhance their quality of life. Due to their extensive network connectivity, IoT devices are used to a significant degree, which raises security concerns. Home automation is a subset of the IoT gadgets that are connected to other devices; items like printers, refrigerators, and thermostats are controlled by Google Assistant and Amazon

[1] *Department of Computer Science and Engineering, Sanskriti University, Mathura, India.*
[2] *Department of Computer Science and Engineering, Sanskriti University, Mathura, India.*
[3] *Department of Computer Science and Technology, Manav Rachna University, Faridabad.*
[4] *Research professor at Universidad Tecnológica de los Anges, Perú.*
[5] *Universidad Nacional Santiago Antúnez de Mayolo, Huaraz, Peru.*
[6] *Department of CSE, Universidad Nacional Santiago Antunez de Mayolo, Huaraz, Peru.*

Alexa, among other artificial intelligence systems.[6,7] Therefore, sending spam emails, joining a botnet, and exposing personal information make it simple to control[8] these devices. They are taken into account while creating IoT devices, as are elements linked to security.[9] Although data transfer is uncountable and IoT device adoption is rising. Massive data exchange and billions of interactions make controlling and tracking tough. The Intrusion Detection System (IDS) system makes use of a variety of hardware and software tools to scan the network and detect malicious activity.

Our most current open-source IoT traffic generating technology is the Internet-of-Things Flock, and it forms the basis of the IoT application builder that serves as the starting point of the proposed procedure. The Internet of Things-Flock tools may simulate both benign and malicious activities of IoT-enabled devices in any given use case. The Internet-of-Things Flock makes it simple to build custom Internet of Things use cases and generate Internet of Things traffic for a wide range of applications, all from a single physical machine. This allows researchers to test their hypotheses in a live, real-world network with a large number of connected devices. Fig. 1 shows the fundamental architecture of the Flock for the Internet of Things.

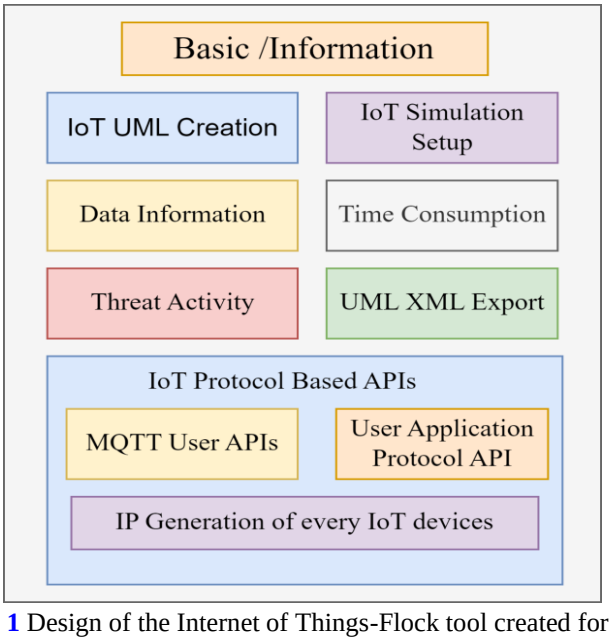


**Fig. 1** Design of the Internet of Things-Flock tool created for the production of IoT traffic.

Consider a scenario where a system or IoT device is subject to Security Information and Event Management (SIEM) control over[10] their operations. To distinguish malicious activity, SIEM blends several origin outcomes and alarm filtering mechanisms.[11] Four types of intrusion prevention systems are used to stop intrusions: host-based, network-based, wireless intrusion, and network behavior analysis. The illegal traffic is successfully predicted using these four categories, which monitor and manage the network infrastructure, wireless connectivity, and software packages. For voice detection in the preventative process, Harris Hawk Sparse Auto-Encoder Networks are used.[12] The Bagging (BG) classifier is recognized as an efficient separator by the machine learning techniques like (five-fold CV) used to create the ConvNet model structure. The effectiveness of the BG classifier is evaluated using a five-fold CV using the deeper subset of features of the chosen ConvNet model. Using a hybrid ConvNet/CNN (Convolutional Neural Networks) learning strategy that incorporates the CNN features, a higher detection rate was attained. The five-fold CV a machine learning technique is extremely trustworthy validation result for the proposed method suggests that (Network Intrusion Detection System) NIDS can be applied in a modern networked computing environment. These techniques analyze network protocol using pre-configured attack patterns in order to successfully forecast intrusion activities. As a result, the threat patterns should be taken into consideration when designing the intrusion detection system. The patterns are divided into misuse intrusion patterns[13] and anomaly intrusion patterns,[14] which can be used to forecast all known threats. In order to decrease intermediate access, this system may occasionally be developed by merging exploitation and anomaly intrusion patterns. Analysis of data, algorithmic evolution, protocol authentication, rule-driven, and machine learning approaches are developed to help identify and terminate intrusion activities.[15] Modern security mechanisms called IDS employ cutting-edge techniques to protect computer networks from invasions that are now taking place or unauthorized access that has already been granted. In order for the IDS to be efficient in cyber security activities, it must also be built to withstand extensive cyber-attacks and real-time testing. Methodology Description Analysis of Statistics[16] in this analysis, the observed behavior is compared to a list of established baselines. Algorithm evolution[17] establishes the implementation route that is used for predicting the average of the model, errors, and distinct behaviors dependent on the situation. Verification of Protocol[18] having checked the protocol field enables the prediction of the suspicious activities. However, the ambiguous protocols are what lead to the false-positive rate. Rules-based[19] by comparing them to the signatures, this method anticipates intrusions. Techniques for machine learning,[20] the intrusions are predicted by assessing the premise using a range of nodes as well as the input mechanism. Numerous strategies are implemented in the Network infrastructure to anticipate intrusion activities. Because the network is educated utilizing sets of experiment or survey, characteristics, and suspicious actions, machine

[7] *Department of CSE, Lovely Professional University, Phagwara, Punjab, India.*
[8] *Department of Computer Science, Koneru Lakshmaaiah education foundation vaddeswaram Guntur, A. P.*
[9] *Department of Computer Engineering and Applications, GLA University, Mathura, India.*
**Email:* kumar.surendra1989@gmail.com (S. Kumar)

learning approaches produce good results and aid in the identification and tracking of attacks, particularly DDoS attacks. These progressive strategies are included to regulate the resistance to attacks by taking the influence of machine learning approaches into consideration throughout the intruder threat analysis procedure.

The main contributions of our research are as follows:

- Due to threats and attacks, privacy and security issues were just an issue in this IoT paradigm.
- Security and privacy issues stemming from penetration threats and attacks afflicted this IoT paradigm.
- The network correctly identifies the standard and threats using training patterns.
- A visual comparison of the effectiveness of the various optimization strategies in identifying fraudulent information over an IoMT healthcare dataset using a variety of evaluation parameters.

The study proposes a novel intrusion detection system, Lion-Salp-Swarm-Optimization Algorithm (LSSOA), which combines metaheuristic algorithms with machine learning techniques to improve intrusion detection and defense against attacks in the IoT-based healthcare environment. The LSSOA framework generates realistic datasets using IoT-Flock software and IoT-23 that evaluates the performance using various metrics, demonstrating its ability to outperform existing intrusion detection systems in detecting and defending against online attacks. The study's contribution lies in its unique approach of combining metaheuristic algorithms with machine learning techniques, enabling the LSSOA framework to detect and defend against previously unseen attacks in the IoT-based healthcare environment. By addressing security and privacy issues posed by the IoT in the healthcare sector, the study provides a promising solution to improve patient safety and security. The LSSOA particularly suitable for intrusion detection method while seeking has the following advantages: first, it generates a circular neighborhood around the solutions, which makes it easier for the detection to approach the prey from numerous sides; second, it enables the solutions to potentially break free of a local optimum; these advantages are listed in order from most advantageous to least advantageous.

## 2. Related work

In Ref. [21] applied the designed method with the help of gradient descent approach to check IoT intrusion movement. This system overcomes its low detection performance and constrained scalability while detecting intruder movement. The PSO-Light approach gathers characteristics from data received and sends information into another support vector algorithm in order to detect fraudulent data. The UNSW-NB15 dataset used for the PSO-Light strategy utilized in this procedure had the highest detection rate for command line, backdoor, and worm activity. Enhancing classification methods or balancing divisions in the supervised learning before the data is loaded into a system for machine learning is one method for handling unbalanced datasets. The latter method is favored because of its wider application. It can be time-consuming and labor-intensive to collect and analyze data, and we frequently have to make up with less-useful facts than we would prefer. Problems are caused by these elements. It's likely that we won't come across enough instances of minority groups to make a strong case. To defend IoT devices from attack attempts, the Passban smart intrusion prevention system was developed.[22]

Passban aids in the detection of malicious traffic, including port scans, flooding assaults, and Secure Shell (SSH) brute-force tries. Because this technology has such a high detection rate, it solves the problems of precision and the prevalence of false positives. To identify the IoT components used in home automation systems that are most vulnerable, parallel supervised authorization was created In Ref. [23] The usual and anomalous behaviors of the IoT device must first be categorized in order to successfully detect attacks like Denial of Service, spoof, replaying assaults. This method can identify multistage assaults with a low percentage of false positives. In order to develop an effective intrusion detection model, the Grasshopper Optimization Algorithm-based Deep Belief Neural Networks (GA-DBN) approach was developed In Ref. [24] This method uses multiple convolution layers and a variable number of genetic algorithm cycles to predict various assault kinds. The enhanced classifier categorizes the attacks with the highest rate of detection for the Network Security Laboratory - Knowledge Discovery in Databases dataset. Additionally, this approach lessens the processing complexity. The term "distributed denial of service," that is used to describe all of these attacks, is a generalization. Online tools called Botnets provide fake traffic for a certain website. DDoS assaults can have severe repercussions for many internet businesses. Savings are made possible by a security that is dependable, trustworthy, and predictable. Operating and capital expenses are decreased because there is not a requirement to use an outside filtering center, hire more information technology personnel, or purchase more bandwidth. The limited access could prevent actual users from finding information or taking the necessary steps. They might have a mark against them. To prepare for anomaly-related intrusion detection, the IoT Backbone system[25] uses the second-tier methodology of categorization and reduction of dimensions. Using linear selection and component evaluation, this method aims to identify remote too local and client root attacks.

The application of web of things is introduced with a two-stage intrusion detection solution In Ref. [26], which uses extracted attributes in combination using K-nearest neighbor and Bayesian Network to detect aberrant activities Sensor-Defined Internet-of-Things (SD-IoT). By this technique, known and unknown SD-IoT attacks are to be recognized. The features are picked using the tuned weights from a randomized forest technique, binary asymmetric mutation, and the bat algorithm. This technique is quite accurate at spotting

abnormal activity and has a low overhead. Service Principal Name (SPN) uses a variety of attack strategies to build its detection system for intrusions.[27]

This method extends the life of the network and reduces IoT infiltration by using a set of parameter values. This system employs probabilistic flow features and composite attack detection techniques to evaluate and defend against malicious assaults. It includes 128 mobile sensor units to monitor network traffic. This study concentrates on the machine learning approaches being used for identification of protocol-related attacks and malicious operations. Using the relationship among factor and efficiency characteristics, malicious acts are removed from the acquired features. As a result, the node guarantees low false positive rates as well as high detection rates. In order to find intrusions, a number of co and multilayer game procedures are developed In Ref. [28] This technology tries to reduce and prevent security-related vulnerabilities by applying multilayered game design. By combining this process with the trust paradigm, the communication technique built on confidence is produced. The system minimizes latency while delivering maximum accuracy, throughput, and security. The modified hashing-based Apriori technique by Azeez *et al*.[29] was implemented upon Apache Machine learning and also is effective in locating and identifying network intrusions by utilizing frequent item sets in mining processes. The suggested technique was evaluated using Knowledge Discovery Datasets (KDD).

In Ref. [30] primarily uses convolution neural networks to detect unauthorized entry into the autonomous Internet of Vehicles. To stop attacks, the Rode Side Unite (RSU) loading behavior is coupled to the data-driven strategy. These characteristics are extracted using the CNN (type of neural network) a technique that protects against RSU assaults. IoT bots that are malicious are identified In Ref. [31] using machine learning algorithms. This strategy lowers the misclassification of risky actions by exploiting appealing network traffic properties. Entropy-based Technique for Order Performance by Similarity to Ideal Solution (TOPSIS) method is used to select the elements that aid in detecting fraudulent activity in the Bot-IoT.[32] Proposes a self-recurrent mathematical model for network intrusion detection based on coefficients of wavelets with bidirectional circular wavelons. To choose the optimum subsets of features and hyper-parameter settings for effective bot identification on the IoT, a Local-Global best Bat Algorithm for Neural Networks (LGBA-NN) approach was proposed In Ref. [33]

In Ref. [34], it was suggested to use a meta-learner powered by machine learning for the last phase of classification along with a stacking ensemble of dense (completely connected) CNNs for malware detection. Classification of Malware with PE headers (ClaMP) dataset was used to assess the methodology. In Ref. [35] a multistage deep learning image recognition technique for identifying network intrusions was introduced. Four-channel graphics are created using the network flow features (R, G, B, A). The machine learning network ResidualNetwork50 is then trained and evaluated using the photos. The ETPC algorithm is introduced, developed on technology, and then contrasted to a number of conventional TPC algorithms In Ref. [36] studied. In Ref. [37] describe an ensemble approach with fusion of data to cope with medical information obtained from BSNs in a fog computing setting. That can provide elevated activity data, a collection of devices has been put together, and the information has been integrated. In this study, the optimum machine learning approach is applied to assess malicious activity in the Web of Things (WoT), utilizing the smallest false positive rate, better detection rate, and minimal complexity. Analysis of the literature approaches revealed that the network was where the majority of the issues were located. The goal is to stop infiltration and threat activities in the IoT by Employing Evolutionary Sparse Convolution Network (ESCNN) methods. If an attacker sends an ICU temperature with an aberrant value using a fake IP, they run the risk of making the ICU vulnerable and suffering severe damage. Fig. 2 displays an illustration of Constrained Application Protocol. Wireless sensor networks' distinctive qualities, such as their high density, low use of electricity, software programming ability, and long-term accuracy, can be credited for their rising popularity. In-depth analyses of important sensor network application metrics, such as lifetime of the network, delivery of packets ratio, energy-efficient delivery, and dead node ratio.[38]

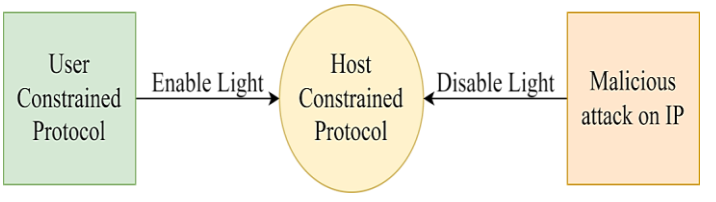


**Fig. 2** Constrained application protocol.

In order to address the difficulties posed by modern software systems like the Internet of Things and artificial intelligence, the study advises creating whole new methodologies.[39] This essay tries to analyze several optimization strategies in-depth, to emphasize their benefits and drawbacks, and to offer suggestions for improvement. The prioritization process must be done in a way that allows for the fastest possible fault detection. Although numerous test case prioritization strategies have been put forth, none have taken into account the danger of omitting test cases. In this study, the multi-criteria optimization problem of test case prioritization or test case subset selection is solved.[40] Using a collection of clinically annotated photographic pictures, already trained CNNs are employed in the current study to identify oral pre-cancerous and malignant lesions and to distinguish them from normal mucosa.[41] Healthcare Big Data Analytics (BDA) is developing into an exciting area for extracting information from very big data sets and has an opportunity to raise healthcare delivery quality while lowering

costs. Medical decision support, monitoring of diseases, and managing population health are just a few of the medical and healthcare fields where BDA has a big impact on how healthcare is delivered. BDA has transformed healthcare and given researchers and clinicians the tools they need to use the data produced by healthcare systems around the world. Through early identification and diagnosis, BDA also helps to reduce adverse occurrences, resulting in safer, more cost-effective procedures.[42] With a high degree of accuracy, ANN (Artificial Neural Network) is crucial in forecasting short-term mortality in patients. Because it automates and makes it simpler to identify patients who are more likely to die, its use in Acute on Chronic Liver Failure (ACLF) patients is promising. The use of ANN in this field has a huge potential to help clinicians make decisions, prioritize individuals in need of urgent liver transplantation,[43] and foretell death and complications.

The study proposes a novel intrusion detection system, Loin-Salp-swarm-Optimization Algorithm (LSSOA), which combines metaheuristic algorithms with machine learning techniques to improve intrusion detection and defense against attacks in the IoT-based healthcare environment. The LSSOA framework generates realistic datasets using IoT-Flock software and evaluates the performance using various metrics, demonstrating its ability to outperform existing intrusion detection systems in detecting and defending against online attacks. The study's contribution lies in its unique approach of combining metaheuristic algorithms with machine learning techniques, enabling the LSSOA framework to detect and defend against previously unseen attacks in the IoT-based healthcare environment. By addressing security and privacy issues posed by the IoT in the healthcare sector, the study provides a promising solution to improve patient safety and security.

## 3. Preliminaries

The suggested framework comprises of five main steps, which are depicted in Fig. 3. One of these processes, which extend from the generation of IoT use case data to it, is the establishment of IoT security parameters for dangerous movement surveillance in IoMT environment. The suggested procedure starts with developing an Internet of Things use case, which is then put into practice to offer real-time IoT traffic.

In order to develop an IoT dataset for optimization algorithms to successfully identify malicious network communication in the fundamental IoT use case, the generated traffic is then gathered. There are various optimization techniques are used like Lion Optimization,[44] Salp-Swarm Optimization,[45] Spider Monkey Optimization,[46] Whale optimization.[47]

### 3.1 Lion optimization

An optimization problem-solving metaheuristic algorithm is called the LOA. Let X = ($x_1$, $x_2$..., $x_n$) be a coordinate in a space with n dimensions, where xi belongs to [L, U] and i = 1, 2, 3,..., N and $x_i$is a number that is real. For each lion i, carry out the chase step:

r1, r2 ~ U (0, 1)

Xj = X[i]

Xbest = X [Xbest]

$$Xnew = Xj + a * (r1 * (Xbest - m * Xj)) \quad (1)$$

X[i] = clip (Xnew, L, U)

$$new\_position = Xj + a * (r1 * (Xbest - m * Xj)) \quad (2)$$

$$new\_position = Xj + a * (r1 * (Xk - m * Xj)) \quad (3)$$

$$new\_position = low\_bound + r1 * (top\_bound - low\_bound) \quad (4)$$

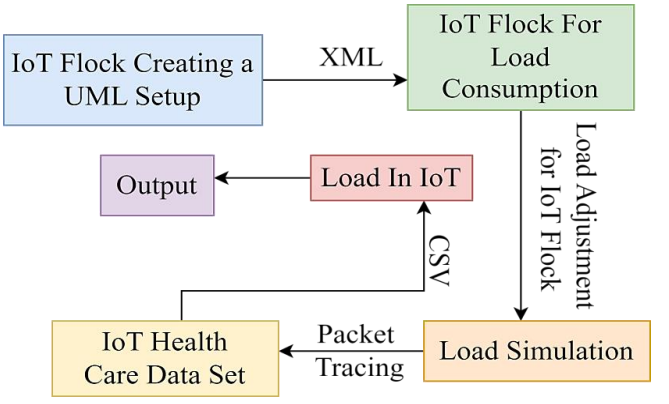


**Fig. 3** Mechanism for IoT security in healthcare research.

In Equations (1-3) shows that Xj is the current position of the lion, X[i] represents the position of lion at a specific index 'i' in the search space (X), Xnew represents the new candidate position that the lion will potentially move to, L and U represents the low bound and top bound of the search space, new_position is used as a variable to represent the updated position of the lion. Xbest is the position of best lion in the pride as well as 'm' and 'a' are the parameters that control the step size and r1 is the random number. In Equation (4), low_bound shows the lower bound of the search space, top_bound shows the upper bound of the search space and r1is the random number between 0 and 1. The Lion Optimization Algorithm[44] (LOA) simulates the behavior of a pride of lions, where each lion have a potential remedy for the performance issue. The algorithm consists of three main steps: chasing, following, and roaming. In the chasing step, each lion updates its position by moving towards the current best lion in the pride. In the following step, each lion updates its position by moving towards a randomly selected lion in the pride. In the roaming step, each lion updates its position by moving randomly within the search space. The algorithm continues for a fixed number of iterations, and the ideal response found so far is returned as the output. The below code implements the LOA to optimize the function f(x) over the search space [-10, 10]. The LOA is a nature-inspired optimization algorithm that mimics the hunting behavior of lions in a pride in Fig. S1. The following steps can be used for developing the LOA algorithm 1:

*Identify the search area*: Set the parameters for the search space's low_bounds and top_bound.
*Initialize the pride:* Create a lion pride with random locations inside the search area. Determine each lion's fitness score in the pride.
*Set the current best lion*: Choose the lion with the highest performance value as the top lion right now. Repeat for a predetermined number of times:
*Chasing step:* Each lion in the pride performs a chasing step, where it updates its position by moving towards the current best lion.
*Following step***:** Each lion in the pride performs a following step, where it updates its position by moving towards a random lion in the pride.
*Roaming step*: Each lion in the pride performs a roaming step, where it updates its position by moving randomly within the area of search.
Update the performance value for each lion in the pride.
Update the current best lion if a lion with a better performance value is found.
Return the position of the best lion found as the solution to the optimization problem. The Python code provided implements the above LOA algorithm with the following parameter values: Pride size: 5, Maximum iterations: 100, m: 5, a: 0.01, the fitness function used is $f(x) = x^2 - 4x + 3$ discuss in algorithm 1.

The LOA begins by randomly initializing a population of lions. In Fig. S2, flow diagram demonstrates how each lion's fitness is subsequently assessed utilizing an objective function. On the basis of its fitness rating, the most capable lion in the community is determined. The best lion is then separated from the other lions, creating two subpopulations within the larger population. In the "hunt" phase, the lions in the subpopulation with the best lion use a search plan based on their present location and speed to look for prey in the solution space. The lions in the subpopulation containing the remaining lions undergo a "roaming" phase where they move randomly in the solution space to explore new regions. The best lion in the community will be provided as an outcome to the optimization issue after a predetermined number of repetitions or until a stopping requirement is satisfied. The stopping criterion can be based on factors such as the number of rounds, the convergence of the answer, or a predetermined threshold. The LOA's output is the best lion found when the algorithm is running, which represents the optimal solution to the optimization problem.

### 3.2 Salp-Swarm Optimization

A population-based metaheuristic optimization technique called Salp-Swarm-Optimization (SSO)[45] was developed in the wake of salps natural behavior. A group of salp, which are imaginary particles that represent possible answers to an optimization issue, is used by the algorithm. Based on forces of attraction and repulsion between them and other salp in the population, the salp move in searching space and alter their positions. Using the provided code and strategy, the SSO algorithm operates as follows:
Specify the goal function that has to be optimized. Set the lower and higher boundaries of the variable xi where i = 1, 2..., N as L denotes the low_bound and U defines the top_bound, respectively, to define the search space. Define the SSO parameters, which govern the ratio of exploration to utilization in the search process. These parameters include the population size, the maximum number of rounds, and the values of the constants c1, c2, and w. Set each salp's position and speed at random when the population of salps is first created within the search space. By assessing the objective function in its current state, determine the level of fitness of each salp in the population. The population member with the lowest fitness value should be the current best salp. The optimum number of iterations should be used: Based on forces of attraction and repulsion with other salps, update the position and velocity of each salp in the population. If the fresh location is outside the search space, use the opposite point method to wrap the salp over to the other side. In the case that xi L, xi = U - (L - xi), and in the event that xi > U, xi = L + (xi - U). Calculate the new position's fitness value. Update the Salp's personal best position if the new fitness is superior to the previous fitness.

$$Vj\,(t+1) = w * Vj\,(t) + c1 * r1 * (Xbest - Xj\,(t)) + c2 * r2 * (Xrand - Xj\,(t)) \quad (5)$$

In Equation (5), each particle's velocity (Vj) at the next step (t+1) is updated based on its current velocity (Vj (t)), a cognitive parameter (c1, c2) that influences its best position, and the global best position (Xbest) in the swarm, Xrand represents the position of a randomly selected particle in the swarm. The update also considers an inertia weight (w) for balancing exploration and exploitation, and random factors (r1, r2) to add stochasticity and promote exploration. The Salp Swarm Optimization (SSO) technique is implemented in the provided code to optimize the function f(x) = x2 - 4x + 3 over the searching space [-10, 10] in Fig. S3. The technique begins by randomly placing and moving a population of salps (a sort of marine animal) inside the search space. Following that, each salp's fitness is assessed using the formula $fitness(x) = x^2 - 4x + 3$.

Every salp in the population goes through two steps in the SSO algorithm: attraction and repulsion. The repulsion step updates the velocity away from neighboring salps to prevent being stuck in local optima, while the attraction step adjusts the velocity of the salp towards its own best position and the global best position. Population_size refers to the number of salps in the population discuss in algorithm 2.

As indicated in the flowchart in Fig. S4, the procedure starts by initializing a group of salps with arbitrary positions and velocities. After the population has been established, the objective function is used to determine each salp's fitness using a user-defined variable, the population's salp count. The degree of accuracy of a given solution is determined by the objective function, which is particular to the issue being solved. Each salp's fitness value is noted. The swarm's leader

 

is then determined to be the salp having the highest fitness value. The other salps must follow this leader's lead in order to find more effective solutions. The speed and position update formulae are then used to update the location and speed of each salp. While the location update equation adjusts the location according to the new velocity, the speed update formula takes into consideration the leader's position and the world's best position. These equations' parameters include random numbers, acceleration coefficients, and inertia weight. The locations are updated, and then they are examined to make sure they do not conflict with any limitations imposed by the issue being fixed. A location that is outside of its range is changed to the closest legitimate position. The fitness of each modified salp is assessed using an objective function once the locations have been changed and verified for boundary restrictions.

The new leader directs other salps towards superior options if a salp with a higher fitness value than the existing leader is discovered. If a termination requirement, such as a maximum number of rounds or convergence, has not been reached, the algorithm checks that it has and, if not, it returns to the position and speed update step. The best salp is given back as the answer to the optimization issue if the termination requirement is satisfied. The Salp Swarm Optimization algorithm, in general, is a metaheuristic optimization method that mimics the behavior of salps in the ocean and can be applied to a variety of optimization issues.

### 3.3 Spider Monkey Optimization

A metaheuristic optimization called Spider Monkey Optimization[46] (SMO) is used to get the least value of the function f(x). The technique establishes a search area that includes the low_bound and top_bound and randomly assigns spots inside this space to a population of spider monkeys. The fitness function, which determines the variables of the supplied function f(x) for that position, is used to assess each spider monkey's position. The algorithm then iterates for a specified number of max_iterations, during which each spider monkey performs three steps: Movement, Diffusion, and Jump.

new_position = Xj + m * (Xbest - Xj) + sigma * r1 (6)

Equation (6) shows that Xj is the current position of the spider monkey, Xbest is the position of the best spider monkey, and r1 is a random number between 0 and 1, ‘sigma’ is another parameter that influences the step size adjustment, new_position defines the updated position.

new_position = Xj + c * (Xk - Xj) + sigma * r2 (7)

Equation (7) shows that Xj is the current position of the spider monkey, Xk is the position of the randomly selected spider monkey, and r2 is a random number between 0 and 1, c is a constant, ‘new_position’ is the updated position.

new_position = Xj + sigma * r3 (8)

Equation (8) shows that Xj is the current position of the spider monkey, and r3 is a random number between 0 and 1, ‘sigma’ is another parameter that influences the step size adjustment, new_position defines the updated position. The Movement step calculates a fresh position for a SMO by taking a weighted average of its present state and the ideal position found so far. The Diffusion step calculates a new position by taking an average weight of the spider monkey's current position and another randomly selected spider monkey's position. The Jump step calculates a new position by adding a random offset to SMO current position. The weights used in each step are controlled by the parameters m and c, and the size of the random offset is controlled by the parameter sigma. After each step, the fitness of the spider monkey at its new position is evaluated, and the top spot discovered so far is updated if necessary. The Graph of Spider-Monkey optimization is shown in Fig. S5. Fitness Function: f(x) = x^2 - 4x + 3, Search space: [-10, 10], Population size: 10, Max iterations: 100. SMO parameters: Movement step: new_position = Xj + m * (Xbest - Xj) + sigma * r1, Diffusion step: new_position = Xj + c * (Xk - Xj) + sigma * r2, Jump step: new_position = Xj + sigma * r3, m = 1, c = 0.7, sigma = 0.1, Fitness function: fitness(x) = x**2 - 4*x + 3, Best spider monkey: best_spider_monkey = min (population, key = lambda spider_monkey: spider_monkey ['fitness']) discussed in algorithm 3.

The Spider Monkey Optimization method, which is a naturally inspired optimization algorithm that imitates the social behavior of spider monkeys in the wild, is shown in the flowchart in Fig. S6. In the initialization phase of the method, groups of spider monkeys are formed at random. After that, each spider monkey's fitness is assessed using an objective function. The activities then choose the population's top spider monkey as the world's best. The algorithm selects a reference spider monkey at random for each spider monkey in the population, then creates a new solution by fusing the spider monkey's location with the original spider monkey's position. The new solution's fitness is assessed, and if it proves to be more advantageous than the spider monkey's present position, the spider monkey's position is revised. If any spider monkey has an improved fitness score than the current world best, the global best is updated. The search range, which establishes the bounds within which the spider monkeys can seek for solutions and the step size, which establishes the greatest movement a spider monkey can make in a single iteration are changed. Each spider monkey's position is subjected to a random disturbance. This cycle is repeated until the stopping conditions, such as a predetermined degree of convergence or an optimal number of iterations are satisfied. The algorithm finally comes to an end, returning the best option as the optimized solution. In conclusion, the Spider Monkey Optimization technique is an iterative algorithm that seeks the global optimum of a specified objective function by emulating the foraging habits of spider monkeys. Each spider monkey in the population serves as a potential answer to the optimization problem in the algorithm. The spider monkeys move towards better solutions by learning from their fellow spider monkeys. The technique can handle both continuous and discontinuous optimization problems, and it has successfully solved a range

of optimization problems in the real world.

### 3.4 Whale optimization

This code is implementing the Whale Optimization Algorithm[47] (WOA) to minimize a fitness function, which is defined as function f(x). A metaheuristic method called the Whale Optimization Algorithm was developed in response to humpback whales' hunting habits. The fitness function accepts just one input variable x and produces a scalar value that measures the solution's quality. The fitness function in this instance is given as $f(x) = x^2 - 4x + 3$. The low and top limits, which are set to -10 and 10, respectively, define the search space. The WOA algorithm explores the search space and identifies the best solution using a population of searching agents. The maximum number of rounds is set to 100, and the number of search agents is set to 5. The three primary steps of the WOA algorithm are spiral update, encircling prey, and searching for prey. On each search agent in the population, each step is carried out. By determining a fresh position for the agent in search based on the location of the best agent in the population, the encircling prey phase is carried out. This action is intended to bring the search agent one step further to the ideal resolution. The search agent's position is modified during the spiral update stage depending on a spiral movement. This step aims to conduct a more systematic search space exploration. The stage of looking for prey involves randomly looking for a new spot in the search space. This step's goal is to introduce some randomness into the search process in order to avoid trapping local optima. The WOA method finishes once the maximum number of iterations has been reached, and the best solution found by the search agents is returned as the best solution in Fig. S7. Simulation Setup: The fitness function is specified as return x**2 - 4*x + 3, the search space's low and top bounds are defined as low_bound and top_bound, respectively, and the WOA variables are defined as num_agents and max_iterations and a, a, and encircling_bounds and [0, 2] respectively.

new_position = best_agent ['position'] - A1 * D1 (9)

Equation (9) shows that A1 is a random coefficient between -A and A, and D1 is the distance between the search agent and the current best agent, best_agent ['position']is the position of the best agent in the optimization process.

D1 = abs (C1 * best_agent ['position'] - agent ['position'] (10)

new_position =

new_position - A2 * math.sin (2 * math.pi * C2) * D1 (11)

In Equation (10), D1 represents the absolute difference between two positions, C1 A constant parameter used for scaling, best_agent ['position']is the position of the best agent in the optimization process, agent['position'] is the position of the current agent.

Equation (11) shows that A2 is a random coefficient between -A and A, C2 is a random coefficient between 0 and 2, and D1 is the same as in the previous step, new_position is the updated position, math.sin () is the mathematical function to perform expression.

new_position = best_agent ['position'] - A3 * D3 (12)

D3 = abs (C1 * best_agent ['position'] - agent ['position']) (13)

Equation (12) shows that A3 is a random coefficient between -A and A, and D3 is the distance between the search agent and the current best agent multiplied by the random coefficient C1, best_agent ['position']is the position of the best agent in the optimization process, agent['position'] is the position of the current agent shows in Equation (13).

As shown in the algorithm 4, the search agents are initialized with random locations throughout the search space: Agents are equal to ['position': random.uniform (low_bound, top_bound) for i in range (num_agents)]. Using the following code, the agent with the lowest fitness value is set as the current best agent: In the formula best_agent = min (agents, key=lambda agent: fitness (agent ['position'])). Consider minimizing the following function:

A population of solutions, referred to as “whales,” is generated at random as part of the WOA algorithm's Initialization step, which is illustrated in Fig. S8. The next stage, Fitness Evaluation, assesses the fitness of each population-level solution. The Most Effective Solution phase finds and saves the best solution so far. A search operator is used in the Exploring step to advance the solutions in the direction of the top solution thus far. This operator promotes a wider search of the solution space and aids in the discovery of novel solutions. An exploitation operator is used in the exploitation process to shape the solutions around the most advantageous one at the time. This operator tries to increase the correctness of the results in the promising parts of the solution space. The Search Space is modified to concentrate on the most promising area of the solution space following each round of Exploration and Exploitation. These phases are repeated until the algorithm meets a stopping requirement, such as completing a predetermined number of iterations or a specified level of solution accuracy. In general, the WOA technique is a metaheuristic optimization method that imitates humpback whale behavior to discover the optimum solution to a problem. The WOA algorithm is able to effectively explore and exploit the solution space to find an ideal solution by iteratively updating and improving a population of solutions.

## 2. Proposed LSSOA methodology

Lion-Salp-Swarm-Optimization Algorithm (LSSOA) is a hybrid optimization method that can be used to prevent DDoS attacks. The technique involves combining linear optimization algorithms with Salp Swarm Optimization (SSO) authentication to lessen the impact of Distributed Denial of Service assaults. Here are the steps to use LSSOA hybrid optimization to prevent DDoS attacks:

Implement Salp Swarm Optimization (SSO) authentication: SSO authentication is a way to authenticate users across multiple systems or applications using a single set of login credentials. Implementing SSO authentication helps prevent DDoS attacks by reducing the number of login requests and authentication attempts made by attackers. Use lion

optimization algorithm: LOA can be used to optimize the usage of available resources, such as bandwidth and server capacity. This algorithm can help identify the optimal configuration for resources to minimize the impact of DDoS attacks.

- *Monitor traffic patterns:* Use traffic monitoring tools to identify traffic patterns and identify potential DDoS attacks. Monitoring traffic patterns can help you identify unusual traffic patterns and prevent DDoS attacks from occurring.
- *Implement rate restriction:* By using rate restriction, the number of queries that can come from an IP address in particular or a number of IP addresses can be restricted. Restricting the number of queries that offenders can make through rate restrictions can help stop DDoS attacks.
- *Implement blacklisting:* Communication from recognized malicious IP addresses or IP address ranges can be blocked using blacklisting. By blocking information from known malicious sources, blacklisting can aid in averting DDoS assaults. Use a Content Delivery Network (CDN) to distribute content over numerous servers, which will lessen the demand on each individual server. By dispersing traffic among several servers, CDNs can also assist in preventing DDoS attacks.

By implementing these steps, you can use LSSOA optimization to prevent DDoS attacks and reduce the impact of any attacks that do occur shows in Fig. 4. Here's an algorithm that combines the Lion Optimization Algorithm (LOA) and Salp Swarm Optimization (SSO) into a hybrid algorithm:

new_position = old_position + (rand () * ((leader_position + global_leader_position)/2 - old_position)) * hybrid_weight (14)

As shown in Equation (14) describes the following terms which includes the new_position: updated position of the individual (lion or salp), old_position: current position of the individual, leader_position: position of the selected leader (the lion with the highest fitness in the population), global_leader_position: position of the global leader (the salp with the highest fitness in the population), rand(): a random number between 0 and 1, hybrid_weight: a weight parameter that controls the step size of the individual towards the midpoint of the leaders.

The computational complexity of this hybrid algorithm (LSSOA) can be loosely approximated as O (Ndl), where N is the population size, d is the search space's dimension, and l is the optimum number of iterations.

This hybrid algorithm in Fig. 5 consists of two metaheuristic algorithms: the Lion Optimization Algorithm (LOA) and the Salp Swarm Optimization (SSO) algorithm. The algorithm starts by randomly initializing a population of lions and salps within the search space. The objective function assesses each lion and salp's fitness value. Leaders are chosen from among the best lions and salps, and the salp with the optimum performance value is chosen as the world leader. The lions are subsequently modified using the LOA update

Algorithm 5

Create a random starting group of lions and salps in the search area.
Identify the population's lions' and salps' fitness levels.
Select the top-performing lions as leaders and the top-performing salps as global leaders
Set the greatest number of iterations and the integration criterion
Set the LOA and SSO weight parameters.
for each iteration in the most iterations possible:
for each lion in the population:
    update the lion's position using the LOA equation:
new_position = old_position + (rand() * (leader_position - old_position)) * LOA_weight
if new_position is outside the search space:
    set the lion's position randomly within the area of search
Appraise the lion's readiness for the fresh role.
if the new position is a better fit than the old one:
modify the lion's position
for each salp in the population:
    update the salp's position using the SSO equation:
new_position = old_position + (rand() * (global_leader_position - old_position)) * SSO_weight
If new_position is outside the search space:
    set the salp's position randomly within the area of search
Assess the salp's fitness in the new position.
Ifthe new position is more fit than the old one:
    update the salp's place
    Calculate the fitness value of both
    select the top-performing lions as leaders and the top-performing salps as global leaders
If the integration criteria met:
Exit from the loop
Output: Return the best solution found

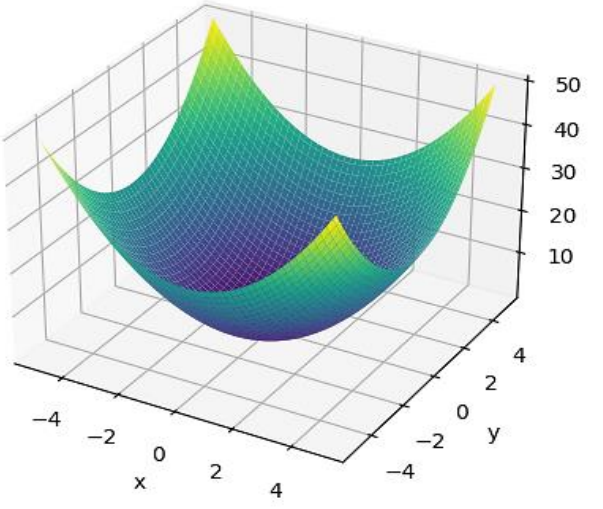


**Fig. 4** Proposed LSSOA optimization to prevent DDoS attacks and reduce impact of attacks.

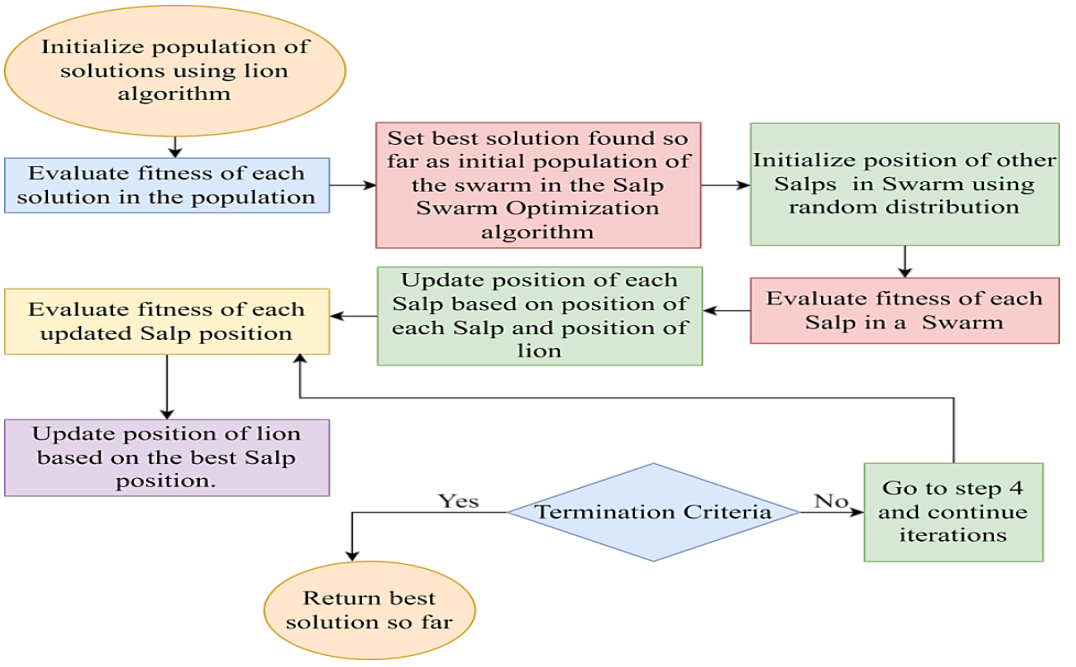


**Fig. 5** Framework of Proposed LSSOA Algorithm to prevent the DDoS attack in IoT-Flock.

equation and the salps are modified using the SSO update equation in every iteration of the algorithm. By evaluating the lions' fitness values' standard deviation, the algorithm tests for integration. The method terminates if the standard deviation drops below a predetermined integration limit. The hybrid approach enhances strength and outcomes of process by combining the qualities of the LOA and SSO methods with other metaheuristics.

## 3. Result and discussion

To assess the efficiency of the suggested approach, the authors employed a system Intel (R) Xeon (R) with configuration Processor E5-2640 (2.50 GHz) and 8 GB of primary memory for the experimental work. We utilized MATLAB to simulate the deployment of 100 SNs at random on a square field with a diameter of 100 x 100m$^2$ and also this dataset responds to the variety of DDoS attacks. When the relevant network components have been constructed, data transmission has started, and the intruder network has started receiving and transmitting data packets, the following the first step is to obtain the data in the data packets flow and messages. For capturing the packets, we utilized Wireshark.[48] The real-time packets are captured and can be filtered and examined using Wireshark,[49] a well-known and open wireless analysis tool. We built a Python utility to monitor data packets flows while capturing the data flow sequentially to retrieve the IoT end level layer characteristics from the Packet Capture files format (.pcap). Research is being done to find ways to defend against DDoS assaults using Hyper Text Transfer Protocol (HTTP) and other conventional application layer protocols.[50] IoT protocol stacks at the application layer are not well protected against DDoS assaults, nevertheless, and little is being done to change that. We developed a Python script[51] to analyze pcap format files and utilize the terminal-oriented version (Tshark) to extract the application and network-level characteristics of Internet of Things. In this research, we extracted network as well as IoT end-level layer server properties from each created a Packet Capture file and stored them into CSV (Comma Separated Values) containing pertinent information traffic classifications. After the traffic was captured, pcap files made up the dataset. The packet capture files were transformed into Comma Separated Values files using a Python code for preprocessing and analysis of the generated dataset. We also utilized Label Encoder to convert the categorical components of the information set, like the service type, into numerical values to simplify later processing. In the end, we analyzed the dataset to look for any missing values. The dataset had a few blanks, which we filled in with 0. Using a division ratio of 70:30, the information set was then arbitrarily divided into groups to be trained and tested, with 30 percent of the dataset selected at random to undergo testing and the other 70% being chosen for training. The four most often used performance metrics for evaluating the efficacy of optimization procedures are here defined:

**3.1 Precision:** refers to detect attacks with accuracy after a security breach has occurred. It gives the ratio of attacks that were accurately anticipated (VP) to those that really occurred (VP + IP). Equation (15) shows the mathematical formulation for Precision.

$$Precision = \frac{\mathrm{VP}}{\mathrm{VP+IP}} \times 100 \quad (15)$$

**3.2 Recall:** describes the system's capacity to accurately identify a botnet attack as soon as it occurs in the network. It is denoted mathematically by Equation (16) shows the mathematical formulation for Recall.

$$Recall = \frac{\mathrm{VN}}{\mathrm{VN+IN}} \times 100 \quad (16)$$

**3.3 Accuracy**: is the system's capacity to distinguish between valid data packets and illegal data packets, classifying the latter as a "assault packet" and the former as a "normal packet." The percentage of accurate predictions made across all samples is given. It is numerically represented as follows in Equation (17) shows the mathematical formulation for Accuracy.

$$Accuracy = \frac{VP+VN}{VP+IN+VN+IP} \times 100 \quad (17)$$

**3.4 F1-Score:** The accuracy ratio of the assessment set is reported for both untargeted and targeted traffic. It is numerically represented as follows in Equation (18) shows the mathematical formulation for F1-Score.

$$F1-Score = 2 \times \frac{Precision*Recall}{Precision+Recall} \quad (18)$$

There are following terms that we have to defined and used above:

*Valid Positive (VP):* The attack flow was correctly identified by the ML model as an attack.

*Valid Negative (VN):* The normal flow was accurately predicted as normal by the ML model.

*Invalid Positive (IP):* The typical flow was misdiagnosed by the ML model as an attack.

*Invalid Negative (IN):* The attack flow was incorrectly classified by the machine learning model as normal.

The VPR is a ratio between all of the real attack samples (VP + IN) and all of the correctly anticipated attacks (VP, or VP + IN). It is represented mathematically as Equation (19):

$$\text{Valid Positive Rate(VPR)} = \frac{VP}{VP+IN} \times 100 \quad (19)$$

The IPR is defined as the ratio of all incorrect attack detection samples (IP) to all legitimate samples (IP + VN). Equation (20) provides a mathematical description of it.

$$Invalid\ Positive\ Rate(INR) = \frac{IP}{IP+VN} \times 100 \quad (20)$$

**Table 1.** To evaluate the effectiveness of various optimization techniques.

| Optimization Methodologies | Accuracy | IPR | INR | Precision | Recall F1-Score |
|---|---|---|---|---|---|
| Lion Optimization | 85.0% | 3.0% | 12.0% | 88.0% | 88.0% 88.0% |
| Salp Swarm Optimization | 81.0% | 6.0% | 18.0% | 84.0% | 82.0% 83.0% |
| Spider Monkey Optimization | 82.0% | 5.0% | 16.0% | 86.0% | 84.0% 85.0% |
| Whale Optimization | 87.0% | 2.0% | 11.0% | 90.0% | 90.0% 90.0% |
| Proposed LSSOA | 99.59% | 7.0% | 18.0% | 99.0% | 98.0% 97.0% |

Table 1 shows the performance metrics for five different optimization techniques used for malicious traffic detection. The metrics include the INR which defines the percentage of positive data that is incorrectly labeled as negative, also known as the Invalid Negative Rate. Precision defines the percentage of actual positive results versus all expected good results. Recall illustrates the percentage of true positives among all actual positives. Each row of the table corresponds to a different optimization technique, allowing for comparison of their performance in terms of these metrics. From the table, we can see that the Whale Optimization Algorithm had the highest accuracy and F1-Score, while the Salp Swarm Optimization had the highest IPR and INR. The Lion Optimization and Hybrid of Lion and Salp Swarm optimization performed well in all metrics.

The Fig. 6 shows the comparison of the performance of five different optimization techniques in detecting malicious traffic over an IoMT dataset. The x-axis represents the different optimization techniques used, which are Lion Optimization, Salp Swarm Optimization, Spider Monkey Optimization, Whale Optimization, and a hybrid of LSSOA. The y-axis represents the evaluation metrics used to evaluate the effectiveness of the various methods. These evaluation metrics include accuracy, Invalid positive rate, Invalid negative rate, precision, F1-Score and recall. Each bar in the diagram represents the value of a particular evaluation metric for a specific optimization technique. For example, the first section in the graph reflects the attained accuracy score by the Lion Optimization technique, while the second section reflects the attained accuracy score by the Salp Swarm Optimization technique, and so on. Overall, the diagram provides a visual comparison of the effectiveness of the various optimization strategies in identifying fraudulent information over an IoMT healthcare dataset using a variety of evaluation parameters presented in Table 2.

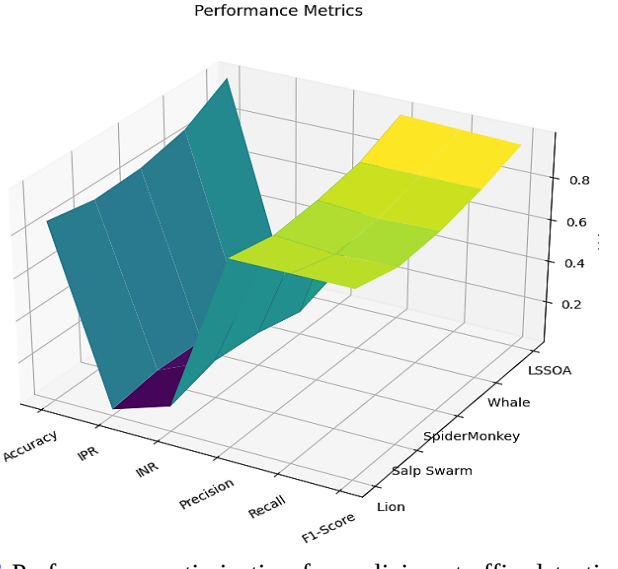


**Fig. 6** Performance optimization for malicious traffic detection over IoMT dataset.

The LSSOA methodology has a wide range of applications in various fields which include Network Security: The LSSOA methodology can be used to prevent Distributed Denial of

**Table 2.** To evaluate the effectiveness of various dataset with accuracy.

| Reference | Dataset | Method | Accuracy Percentage |
|---|---|---|---|
| Patil and Kshirsagar[52] | CICIDS2017 | information gain and ranker algorithm | 87.44 |
| Lima Filho d. *et al*.[53] | ISCXIDS2012 | RFECV and Random forest | 99.42 |
| Saeed, A.A.[54] | Balanced dataset | B-PSO with DT | 99.52 |
| Bista *et al*.[55] | CAIDA UCSD DRAPA2000 | Heuristic clustering algorithm and Nave-Bayesian classifier | 99.45% , 86.73% |
| Arivudainambi *et al*.[56] | NSL- KDD cup | Lion optimization algorithm + Convolutional neural network | 98.2% |
| Sreeram *et al*.[57] | CAIDA | bio-inspired bat algorithm | 94.8% |
| Mohammadi, S. [58] | CICIDS | HMGOGA + Random Forest | 98.3496% |
| Hazman, C. [59] | IoT-23 | SwiftIDS +LightGBM | 98.7% |
| Proposed Method | IoT-Flock | LSSOA | 99.59% |

Service (DDoS) attacks and enhance network security. Organizations that rely on online platforms for their business operations, such as e-commerce sites, financial institutions, and government agencies, can use the LSSOA methodology to protect their networks, resources, and customers from such attacks. Resource Allocation: The LSSOA methodology can optimize the utilization of available resources, such as bandwidth and server capacity, by identifying the optimal configuration for resources to minimize the impact of DDoS attacks. Optimization: The LSSOA methodology can be used to optimize various processes and systems, such as supply chain management, transportation, and logistics. The IoT-23 dataset used to enhance the experimental capability to clearly mentioning how framework or proposed approach feasibly works.[60] IoT-23 consists of twenty malware captures obtained from a variety of different IoT devices in addition to three benign anomaly captures. The experimental evaluation of our method is carried out on a Kaggle computer that has a 64-bit operating system, Jupyter Lab and Python 3.9.7 are utilized in the construction of the model, in addition to the modules known as pandas, numpy, and sklearn. The classification of IDS is an extremely important matter. In addition, the dataset that is used for model training and testing is what determines the optimal performance factors of any classifier. The scope of the LSSOA methodology includes the Hybrid Optimization: The LSSOA methodology combines the strengths of two different optimization techniques, the Lion Optimization Algorithm (LOA) and Salp Swarm Optimization (SSO), to achieve better results than using either technique alone. Multiple Variables: The LSSOA methodology can optimize multiple variables simultaneously, making it suitable for complex problems that involve numerous parameters. Scalability: The LSSOA methodology is scalable and can be applied to small-scale as well as large-scale networks and systems. The limitation of the LSSOA methodology includes Computational Resources: The LSSOA methodology may require significant computational resources and time, particularly when applied to large-scale networks with numerous nodes. Accuracy of Traffic Monitoring: The effectiveness of the LSSOA methodology depends on the accuracy of the traffic monitoring tools and the ability to detect unusual traffic patterns. Scope of Application: The LSSOA methodology may not be suitable for all optimization problems and may have limitations in certain fields or applications.

## 6. Conclusion

Due to the fast growth of Internet of Things technology, researchers and professionals are focusing their efforts on the growth of Internet of Medical Things (IoMT) environments. Numerous IoMT solutions have recently been offered, but these systems still contain security vulnerabilities. The safety of IoMT networks is crucial, as any kind of security flaw or online attack on IoMT environment leads to substantial effect on mankind's existence. In this paper, we presented a methodology for the development of Internet of Things security parameters that can detect malicious activities in IoMT environments. To extract only the most relevant characteristics from a dataset, a feature selection strategy is employed. About 75% of the aforementioned dataset's 77 characteristics have been eradicated because they were deemed unnecessary. Utilizing multiple cost functions, the LOA algorithm employs particular characteristics. In light of numerous LOA flaws, such as slow convergence and getting stuck in local minimums (termed LSSOA), this method is updated and then combined with a genetic algorithm. The results of the tests illustrate how effectively the suggested framework is applicable to create trustworthy, IoT security parameters. In addition, the created dataset and suggested approach enable the researchers to implement the recommended strategy for developing IoT security parameters, particularly for IoMT environment. Additionally, the proposed architecture makes it easy for researchers to generate data for fresh Internet of Things applications to establish artificial intelligence-based safety measures for these new IoT

use cases.

## Conflict of Interest
There is no conflict of interest.

## Supporting Information
Applicable, available at

## References

[1] S. Sundar, A. Singh, A swarm intelligence approach to the early/tardy scheduling problem, *Swarm and Evolutionary Computation*, 2012, **4**, 25-32, doi: 10.1016/j.swevo.2011.12.002.
[2] J. Layegh, F. Jolai, A memetic algorithm for minimizing the total weighted completion time on a single machine under linear deterioration, *Applied Mathematical Modelling*, 2010, **34**, 2910-2925, doi: 10.1016/j.apm.2010.01.002.
[3] R, Soltani, Two robust meta-heuristics for scheduling multiple job classes on a single machine with multiple criteria, *Expert Systems With Applications*, 2010, **37**, 5951-5959, doi: 10.1016/j.eswa.2010.02.009.
[4] J, Behnamian, Minimizing makespan on a three-machine flowshop batch scheduling problem with transportation using genetic algorithm, *Applied Soft Computing*, 2012, **12**, 768-777, doi: 10.1016/j.asoc.2011.10.015.
[5] S. M. Goldansaz, F. Jolai, A. H. Zahedi Anaraki, A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open shop, *Applied Mathematical Modelling*, 2013, **37**, 9603-9616, doi: 10.1016/j.apm.2013.05.002.
[6] J. Senthilnath, S. N. Omkar, V. Mani, Clustering using firefly algorithm: performance study, *Swarm and Evolutionary Computation*, 2011, **1**, 164-171, doi: 10.1016/j.swevo.2011.06.003.
[7] S. Jagannath, Nanda, A survey on nature inspired metaheuristic algorithms for partitional clustering, *Swarm and Evolutionary Computation*, 2014, **16**, 1-18, doi: 10.1016/j.swevo.2013.11.003.
[8] R. Panda, M. K. Naik, B. K. Panigrahi, Face recognition using bacterial foraging strategy, *Swarm and Evolutionary Computation*, 2011, **1**, 138-146, doi: 10.1016/j.swevo.2011.06.001.
[9] G. Fornarelli, A. Giaquinto, An unsupervised multi-swarm clustering technique for image segmentation, *Swarm and Evolutionary Computation*, 2013, **11**, 31-45, doi: 10.1016/j.swevo.2013.02.002.
[10] R. Oftadeh, M. J. Mahjoob, M. Shariatpanahi, A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search, *Computers & Mathematics With Applications*, 2010, **60**, 2087-2098, doi: 10.1016/j.camwa.2010.07.049.
[11] V. Bhargava, S. E. K. Fateen, A. Bonilla-Petriciolet, Cuckoo Search: A new nature-inspired optimization method for phase equilibrium calculations, *Fluid Phase Equilibria*, 2013, **337**, 191-200, doi: 10.1016/j.fluid.2012.09.018.
[12] Y.-J. Zheng, Water wave optimization: a new nature-inspired metaheuristic, *Computers & Operations Research*, 2015, **55**, 1-11, doi: 10.1016/j.cor.2014.10.008.
[13] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.
[14] J. Doyne Farmer, N. H. Packard, A. S. Perelson, The immune system, adaptation, and machine learning, *Physica D: Nonlinear Phenomena*, 1986, **22**, 187-204, doi: 10.1016/0167-2789(86)90240-x.
[15] M. Dorigo, Optimization, learning and natural algorithms. Ph. D. Thesis, Politecnico di Milano, 1992.
[16] R. Eberhart, J. A. Kennedy, new optimizer using particle swarm theory, *IEEE City*, 1995, doi: 10.1109/MHS.1995.494215
[17] Abbass, H. A. MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach, *IEEE*, City, 2001, doi: 10.1109/CEC.2001.934391
[18] K. M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Systems Magazine*, 2002, **22**, 52-67, doi: 10.1109/MCS.2002.1004010
[19] M. M. Eusuff, K. E. Lansey, Optimization of water distribution network design using the shuffled frog leaping algorithm, *Journal of Water Resources Planning and Management*, 2003, **129**, 210-225, doi: 10.1061/(asce)0733-9496(2003)129: 3(210).
[20] S.-C. Chu, Tsai, P.-W. and Pan, J.-S. Cat swarm optimization. *Springer, City*, 2006, doi: 10.1007/978-3-540-36668-3-94
[21] R. A. Mehrabian, A novel numerical optimization algorithm inspired from weed colonization, *Ecological Informatics*, 2006, **1**, 355-366, doi: 10.1016/j.ecoinf.2006.07.003.
[22] A. Mucherino, O. Seref, O. Seref, O. E. Kundakcioglu, P. Pardalos, Monkey search: a novel metaheuristic search for global optimization, *AIP Conference Proceedings*, *American Institute of Physics*, 2007, doi: 10.1063/1.2817338.
[23] F.-C. Yang, Y.-P. Wang, Water flow-like algorithm for object grouping problems, *Journal of the Chinese Institute of Industrial Engineers*, 2007, **24**, 475-488, doi: 10.1080/10170660709509062.
[24] S. Dan, Biogeography-based optimization, *IEEE Transactions on Evolutionary Computation*, 2008, **12**, 702-713, doi: 10.1109/TEVC.2008.919004.
[25] C. J. A. Bastos Filho, F. B. de Lima Neto, A. J. C. C. Lins, A. I. S. Nascimento, M. P. Lima, A novel search algorithm based on fish school behavior, 2008 IEEE International Conference on Systems, Man and Cybernetics, October 12-15, 2008, Singapore. IEEE, 2009, 2646-2651, doi: 10.1109/ICSMC.2008.4811695.
[26] P. Chauhan, M. Atulkar, An efficient centralized DDoS attack detection approach for Software Defined Internet of Things, *The Journal of Supercomputing*, 2023, **79**, 10386-10422, doi: 10.1007/s11227-023-05072-y.
[27] R. Rajabioun, Cuckoo optimization algorithm, *Applied Soft Computing*, 2011, **11**, 5508-5518, doi: 10.1016/j.asoc.2011.05.008.
[28] X. -S. Yang, Firefly algorithms for multimodal optimization, *Springer City,* 2009, doi: 10.1007/978-3-642-04944-6-14.
[29] S. Yang, J. Jianjun, G. Yang, A dolphin partner optimization, *WRI Global Congress on Intelligent Systems*, 2009, **1**, 124-128, doi: 10.1109/GCIS.2009.464.

[30] A. Kaveh, N. Farhoudi, A new optimization method: dolphin echolocation, *Advances in Engineering Software*, 2013, **59**, 53-70, doi: 10.1016/j.advengsoft.2013.03.004.
[31] X. S. Yang, Flower pollination algorithm for global optimization. *Springer City*, 2012, doi: 10.1007/978-3-642-32894-7-27.
[32] A. Hossein, Gandomi, Krill herd: A new bio-inspired optimization algorithm, *Communications in Nonlinear Science and Numerical Simulation*, 2012, **17**, 4831-4845, doi: 10.1016/j.cnsns.2012.05.010.
[33] R. Tang, S. Fong, X.-S. Yang, S. Deb, Wolf search algorithm with ephemeral memory. Seventh International Conference on Digital Information Management (ICDIM 2012), 2012, Macau, Macao, China. IEEE, 2012, 165-172, doi: 10.1109/ICDIM.2012.6360147.
[34] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Advances in Engineering Software*, 2014, **69**, 46-61, doi: 10.1016/j.advengsoft.2013.12.007.
[35] H. Eskandar, A. Sadollah, A. Bahreininejad, M. Hamdi, Water cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems, *Computers & Structures*, 2012, **110-111**, 151-166, doi: 10.1016/j.compstruc.2012.07.010.
[36] E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez-Cisneros, A swarm optimization algorithm inspired in the behavior of the social-spider, *Expert Systems With Applications*, 2013, **40**, 6374-6384, doi: 10.1016/j.eswa.2013.05.041.
[37] M. Ghaemi, M.-R. Feizi-Derakhshi, Forest optimization algorithm, *Expert Systems With Applications*, 2014, **41**, 6676-6687, doi: 10.1016/j.eswa.2014.05.009.
[38] C. P. Verma, Enhancing parameters of LEACH protocol for efficient routing in wireless sensor networks, *Journal of Computers, Mechanical and Management*, 2023, **2**, 26-31, doi: 10.57159/gadl.jcmm.2.1.23040.
[39] S. Kumar, Reviewing software testing models and optimization techniques: an analysis of efficiency and advancement needs, *Journal of Computers, Mechanical and Management*, 2023, **2**, 32-46, doi: 10.57159/gadl.jcmm.2.1.23041.
[40] S. Deepa, Metaheuristics for multi criteria test case prioritization for regression testing, *Journal of Computers, Mechanical and Management*, 2022, **1**, 38-47, doi: 10.57159/gadl.jcmm.1.1.22015.
[41] D. Sharma, V. Kudva, V. Patil, A. Kudva, R. S. Bhat, A convolutional neural network based deep learning algorithm for identification of oral precancerous and cancerous lesion and differentiation from normal mucosa: a retrospective study, *Engineered Science*, 2022, **18**, 278-287, doi: 10.30919/es8d663.
[42] N. Naik, Y. Rallapalli, M. Krishna, A. S. Vellara, D. KShetty, V. Patil, B. Z. Hameed, R. Paul, N. Prabhu, P. B. Rai, Demystifying the Advancements of Big Data Analytics in Medical Diagnosis: An Overview, *Engineered Science*, 2021, **19**, 42-58, doi: 10.30919/es8d580.
[43] B. Musunuri, S. Shetty, D. K. Shetty, M. K. Vanahalli, A. Pradhan, N. Naik, R. Paul, R. Acute-on-chronic liver failure mortality prediction using an artificial neural network, *Engineered Science*, 2021, **15**, 187-196, doi: 10.30919/es8d515.
[44] M. Yazdani, F. Jolai, Lion Optimization Algorithm (LOA): a nature-inspired metaheuristic algorithm, *Journal of Computational Design and Engineering*, 2016, **3**, 24-36, doi: 10.1016/j.jcde.2015.06.003.
[45] R. Abu Khurma, I. Almomani, I. Aljarah, IoT botnet detection using salp swarm and ant lion hybrid optimization model, Symmetry, 2021, **13**, 1377, doi: 10.3390/sym13081377.
[46] E. Sandhya, A. Kumarappan, Enhancing the performance of an intrusion detection system using spider monkey optimization in IoT, *International Journal of Intelligent Engineering and Systems*, 2021, **14**, 30-39, doi: 10.22266/ijies2021.1231.04.
[47] M. S. Aliabadi, A. Jalalian, Detection of attacks in the Internet of Things with the feature selection approach based on the whale optimization algorithm and learning by majority voting, 2023, doi: 10.21203/rs.3.rs-2424464/v2.
[48] Wireshark. City. https://www.chappell-university.com/post/geoip-mapping-in-wireshark.
[49] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, G. A. Shah, IoT-flock: an open-source framework for IoT traffic generation. 2020 International Conference on Emerging Trends in Smart Technologies (ICETST). March 26-27, 2020, Karachi, Pakistan. IEEE, 2020, 1-6, doi: 10.1109/ICETST49965.2020.9080732.
[50] A. Praseed, P. S. Thilagam, DDoS attacks at the application layer: challenges and research perspectives for safeguarding web applications, *IEEE Communications Surveys & Tutorials*, 2019, **21**, 661-685, doi: 10.1109/COMST.2018.2870658.
[51] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia, E. Zdravevski, A framework for malicious traffic detection in IoT healthcare environment, *Sensors*, 2021, **21**, 3025, doi: 10.3390/s21093025.
[52] A. Patil, D. Kshirsagar, Towards feature selection for detection of DDoS attack. Advances in Intelligent Systems and Computing. *Singapore: Springer Singapore*, 2019, 215-223, doi: 10.1007/978-981-32-9515-5_21.
[53] F. S. de Lima Filho, F. A. F. Silveira, A. de Medeiros Brito Jr, G. Vargas-Solar, L. F. Silveira, Smart detection: an online approach for DoS/DDoS attack detection using machine learning, *Security and Communication Networks*, 2019, **2019**, 1-15, doi: 10.1155/2019/1574749.
[54] A. Abubakr Saeed, N. G. M. Jameel, Intelligent feature selection using particle swarm optimization algorithm with a decision tree for DDoS attack detection, *International Journal of Advances in Intelligent Informatics*, 2021, **7**, 37, doi: 10.26555/ijain.v7i1.553.
[55] S. Bista, R. Chitrakar, DDoS attack detection using heuristics clustering algorithm and Naïve Bayes classification, *Journal of Information Security*, 2018, **9**, 33-44, doi: 10.4236/jis.2018.91004.
[56] D. Arivudainambi, K. A. Varun Kumar, S. Sibi Chakkaravarthy, LION IDS: A meta-heuristics approach to detect DDoS attacks against Software-Defined Networks, *Neural Computing and Applications*, 2019, **31**, 1491-1501, doi:

 

10.1007/s00521-018-3383-7.
[57] I. Sreeram, V. P. K. Vuppala, HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm, *Applied Computing and Informatics*, 2019, **15**, 59-66, doi: 10.1016/j.aci.2017.10.003.
[58] S. Mohammadi, M. Babagoli, A hybrid modified grasshopper optimization algorithm and genetic algorithm to detect and prevent DDoS attacks, *International Journal of Engineering*, 2021, **34**, 811-824, doi: 10.5829/ije.2021.34.04a.07.
[59] C. Hazman, A. Guezzaz, S. Benkirane, M. Azrour, lIDS-SIoEL: intrusion detection framework for IoT-based smart environments security using ensemble learning, *Cluster Computing*, 2022, **1-15**, doi: 10.1007/s10586-022-03810-0.
[60] Stratosphere. IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic - Stratosphere Laboratory Datasets. https://www.stratosphereips.org/datasets-iot23.