

Article

A Distributed Island-Based Feature Selection Framework for IoT Intrusion Detection Systems

Jamil Al-Sawwa ^{1,*}  and Aws A. Magableh ^{2,*} ¹ Computer Science Department, Tafila Technical University, Tafila 66110, Jordan² Department of Software Engineering, Prince Sultan University, Riyadh 11586, Saudi Arabia

* Correspondence: jalsawwa@ttu.edu.jo (J.A.-S.); amagable@psu.edu.sa (A.A.M.)

Abstract

The widespread deployment of Internet of Things (IoT) environments has led to an increasing number of cyberattacks, highlighting the need for efficient and accurate intrusion detection systems. Over the last few decades, Intrusion Detection Systems (IDSs) have been proposed to tackle this challenge. However, IDSs face challenges when dealing with high-dimensional IoT data that include redundant or irrelevant features, which can lead to increased false positives and decreased detection performance. An optimization-based IDS framework is one of the solutions for reducing data dimensionality and improving detection accuracy. However, the serial implementation of this type of IDS suffers from high computational time as the volume of data and its dimensionality increase. In this paper, we propose a scalable distributed island-based feature selection IDS using Apache Spark (version 3.5.6), called DISFS-IDS. DISFS-IDS follows a two-level partitioning strategy—data and population—to distribute the workload across worker nodes in order to identify the most informative features while achieving high detection accuracy. Using binary and multiclass IoT datasets, the experimental results demonstrate that DISFS-IDS achieves statistically comparable detection performance to the serial SFOA-based IDS while selecting a smaller subset of features. Moreover, DISFS-IDS provides effective feature reduction and competitive or superior performance compared with Spark-based filter feature selection methods. In the scalability analysis, DISFS-IDS achieves significant speedup as the number of islands increases while maintaining high parallel efficiency.

Keywords: Internet of Things (IoT); intrusion detection system (IDS); optimization algorithm; feature selection; big data; Apache Spark

1. Introduction

Recently, the Internet of Things (IoT) has increasingly been integrated into many domains, such as healthcare, smart homes, and transportation. It allows heterogeneous IoT devices from various vendors to communicate and operate together efficiently. Despite these advantages, the rapid deployment of IoT devices introduces security vulnerabilities because most of these devices have limited computational resources and weak security configurations. Therefore, detecting attacks in real time poses a challenge for researchers to design smart and adaptive Intrusion Detection Systems (IDSs) [1].

IDSs are commonly classified as anomaly-based or signature-based IDS. Anomaly-based IDS compares the normal behavior baseline with the current traffic to detect any anomalous traffic. In a signature-based IDS, the system adapts stored signature patterns to



Received: 1 April 2026

Revised: 11 June 2026

Accepted: 22 June 2026

Published: 27 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

detect any malicious traffic. These types of IDSs encounter challenges when dealing with the rapidly changing nature of IoT-based threats.

Recently, various distributed frameworks, including federated intrusion detection and Spark-based cybersecurity solutions, have been proposed to address scalability and large-scale data processing challenges [2–5]. However, to the best of our knowledge, limited research has investigated distributed wrapper-based feature selection using metaheuristic optimization on Apache Spark for intrusion detection systems. This highlights a significant research gap and motivates the development of a distributed metaheuristic-based feature selection framework.

In the last decade, several studies have employed nature-inspired optimization techniques as wrapper-based feature selection (FS) methods across various domains in a serial manner [6–9]. The robustness and efficiency of these approaches have encouraged researchers to design and develop IDS models by integrating machine learning algorithms with nature-inspired optimization techniques to enhance IDS capabilities [10–14]. This integration demonstrates effective performance in improving detection accuracy while reducing data dimensionality.

Despite these advantages, most existing optimization-based feature selection methods for intrusion detection systems are implemented in a serial manner [10–14], resulting in increased computational overhead as IoT data grow in size and complexity. Consequently, scalability becomes a significant challenge when these methods are applied to large-scale IoT traffic datasets. In addition, there is a notable lack of approaches that integrate optimization-based feature selection methods for intrusion detection systems with distributed computing environments, such as Apache Spark, to enable scalable and efficient processing of large-scale data.

To address the above limitation, we propose a Distributed Island-Based Feature Selection framework for Intrusion Detection Systems (DISFS-IDS) using Apache Spark. DISFS-IDS is designed to efficiently process large-scale IoT datasets while maintaining high detection performance and reducing feature dimensionality. The framework adopts a distributed independent island architecture and a two-level partitioning mechanism. In this mechanism, both the training data and the global population are partitioned across islands, enhancing parallel efficiency and maintaining population diversity throughout the optimization process.

For feature selection, DISFS-IDS employs the Starfish Optimization Algorithm (SFOA), a recent bio-inspired metaheuristic introduced in 2025 by Zhong et al. [15]. SFOA was selected due to its demonstrated effectiveness in solving global optimization problems and its competitive performance against more than 100 optimization algorithms reported in the original study. The main contributions of this paper are summarized as follows:

- A Distributed Island-based Feature Selection framework (DISFS-IDS) using Apache Spark is proposed for intrusion detection systems to effectively process large-scale IoT datasets across distributed computing nodes.
- An independent island architecture is proposed to minimize synchronization and communication overhead in distributed environments.
- A two-level partitioning strategy is proposed, in which both the dataset and the optimization population are partitioned across islands. Each island independently performs feature selection, thereby improving scalability and reducing computational time.
- An adaptive threshold aggregation strategy is proposed to extract the final set of informative features. The threshold is dynamically determined based on the mean and standard deviation of the normalized feature scores, enabling the automatic identification of the most relevant features.

- The proposed IDS framework is designed to be classifier-independent and optimization-independent, allowing the use of different machine learning classifiers and optimization algorithms within the same distributed framework.
- DISFS-IDS is evaluated using binary and multiclass datasets covering various IoT attack scenarios. The experimental results are reported in terms of execution time, number of selected features, and detection performance metrics.

The rest of this paper is organized as follows. In Section 2, we review previous studies on intrusion detection, focusing on optimization-based feature selection and detection improvement. Section 3 introduces the original SFOA. The proposed distributed island-based feature selection IDS is presented in Section 4. Section 5 presents the datasets, experimental setup, and evaluation metrics. Experimental results are analyzed and discussed in Section 6. Finally, Section 7 provides the conclusions and suggests possible directions for future work.

2. Related Works

This section presents recent related research works that use the adopted metaheuristic optimization algorithms as wrapper-based feature selection techniques in the intrusion detection systems. The motivation behind the use and adoption of metaheuristic optimization algorithms by researchers is primarily attributed to their ability to effectively handle complex optimization problems, as well as their simplicity of implementation [16].

Almseidin et al. [10] developed an intrusion detection system (IDS) that integrates Artificial Rabbit Optimization (ARO)-based feature selection with a Decision Tree classifier to identify botnet attacks in IoT environments. The proposed IDS was evaluated using the N-BaIoT dataset, which includes various botnet attack scenarios. The findings indicated that the model achieved competitive performance compared to filter-based methods in terms of both detection effectiveness and feature space reduction.

Elsaid et al. [11] proposed two intrusion detection approaches that combine Grey Wolf Optimization (GWO) with deep learning models, specifically GRU and LSTM, to reduce feature space and improve detection performance. Using NSL-KDD and UNSW-NB15 datasets, the experimental results demonstrate that the suggested model obtained 94% accuracy in anomaly detection and 92% in signature-based detection, showing improved results compared to previous studies.

Maazalahi and Hosseini [12] investigated the impact of Botnet attacks in IoT-based smart environments; they have highlighted their potential consequences on system security. They proposed an intrusion detection system that combines machine learning and metaheuristic optimization to improve the detection of different types of botnet attacks. Their approach was relying on the Sailfish Optimizer (SFO) to be employed as a wrapper-based feature selection method, and its output was used to initialize the Whale Optimization Algorithm (WOA) with promising candidate solutions. Furthermore, they utilized Particle Swarm Optimization (PSO) integrated with K-means clustering as the detection module. They used the BoT-IoT dataset, and the results of their proposed IDS achieved accuracies of 0.998 under controlled conditions and 0.955 under more realistic conditions, demonstrating its effectiveness across different evaluation settings.

In addition, Al Mazroa and his research team [13] examined the integration of binary GWO as a wrapper-based feature selection method within a deep learning framework. The optimizer was utilized to select the most important features, which were then fed into a deep neural network model called the Elman Spike Neural Network (ESNN). ESNN acted as the detection model, enabling the identification of different types of cyberattacks. To further improve performance, the researchers employed the Archimedes Optimization Algorithm (AOA) for hyperparameter tuning. Their proposed approach was evaluated on selected

datasets, achieving an accuracy of 99.12% on NSL-KDD and 99.36% on CIC-IDS2017 in a cyber–physical systems setting.

The authors [14] proposed a hybrid intrusion detection system that combines the Grey Wolf Optimizer (GWO) and the Quantum Binary Bat (QBB) algorithm to select a subset of features. The proposed approach employs several classifiers during the wrapper stage, including Naive Bayes (NB), K-Nearest Neighbor (KNN), and Random Forest (RF). Experimental results on the UNSW-NB15 dataset demonstrated that the proposed IDS model achieved a detection accuracy of 98.5% using the RF classifier.

In [17], the authors proposed a hybrid wrapper-based feature selection method to improve detection accuracy and reduce false alarm rates by selecting optimal feature subsets. In this work, the Grey Wolf Optimizer (GWO) and the Lion Optimization Algorithm (LOA) were used to select optimal feature subsets for a Random Forest (RF) classifier. Experimental results on the CICDDoS2019 and UNSW-NB15 datasets demonstrated the effectiveness of the proposed method in detecting different types of cyberattacks.

Pourardebil et al. [18] introduced a hybrid machine learning-based feature selection framework for intrusion detection in distributed IoT environments. The framework consists of four phases: data preprocessing, filter-based feature ranking, feature subset optimization, and classification. In this framework, three filter-based feature ranking methods and fuzzy TOPSIS were first employed to generate a consensus set of top-ranked features. Then, a novel Discrete Grey Wolf Optimization (DGWO) algorithm was used to optimize the selected feature subset. Using the NSL-KDD and UNSW-NB15 datasets, the experimental results demonstrated that the proposed approach outperformed several state-of-the-art methods in terms of accuracy, precision, recall, F-score, and specificity.

Rani and Reddy [19] explored the limitations and challenges of existing intrusion detection systems, including the lack of generalization capability. To address this issue, they proposed a hybrid intrusion detection framework that combines deep learning with metaheuristic optimization techniques. They begin with data preprocessing, including duplicate removal and normalization. After that, an autoencoder is used to extract deep features. Their approach then applies feature selection based on an improved Chimp Optimization Algorithm (IChOA). Finally, the selected features are fed into DINet, a deep learning model that integrates Deep Temporal Convolution Networks (DTCNs) and Gated Recurrent Units (GRUs). Their experimental results show that the proposed model achieves high accuracy and precision rates of 97%.

Furthermore, Ogunseyi and Thiyagarajan [20] proposed an intrusion detection model based on a deep learning approach that integrates both statistical methods and metaheuristic algorithms to select the most relevant features. Their model is built on a Long Short-Term Memory (LSTM) network and was evaluated using two IoT datasets (NF-BoT-IoT-v2 and IoTID20). The results revealed that the achieved accuracy rates were approximately 98.42% on the NF-BoT-IoT-v2 dataset and 89.54% on the IoTID20 dataset.

In the same context, Benmalek and Seddiki [21] proposed an anomaly-based intrusion detection system using various machine learning and deep learning techniques. In their model, Particle Swarm Optimization (PSO) is employed for feature selection. Using the RTIoT2022 dataset, the results indicated that the combination of categorical boosting and PSO achieved higher performance compared to other methods.

The authors in [22] proposed a dynamic feature selection approach based on the Enhanced Gorilla Troops Optimizer (EGTO) to identify the most relevant features for intrusion detection systems. The proposed method was applied and evaluated on intrusion detection datasets. The experimental results demonstrated that the method outperformed traditional approaches in terms of both feature selection quality and classification performance.

Moreover, the authors in [23] introduced a new wrapper-based feature selection method using the Scatter Search (SS) algorithm. This proposed method is integrated with a support vector machine to detect cyberattacks. In this work, the NSL-KDD dataset was used to evaluate the suggested model, and the experimental results revealed that their approach achieved a high detection accuracy of 99%, a low false alarm rate of 0.02, and selected only 17 out of 41 original features.

The authors in [24] proposed an intrusion detection framework that integrates Linear Discriminant Analysis (LDA) for dimensionality reduction and an Enhanced Whale Optimization Algorithm (EWOA) for feature selection to improve detection accuracy and reduce the false alarm rate. The framework employs an ensemble classifier based on Random Forest (RF) and XGBoost. Using the NSL-KDD dataset, the results demonstrated that the proposed method achieved a classification accuracy of 96.08%, a detection rate of 94.03%, and a false alarm rate of 0.01%.

In [25], the authors investigated the impact of feature selection techniques on improving the performance of intrusion detection systems using Deep Neural Networks (DNNs). Several feature selection methods were evaluated, including correlation-based feature selection and information gain. Using the oversampled NSL-KDD dataset, the combination of SHAP and correlation analysis achieved the best performance, attaining an F1-score of 98.9% while reducing the number of selected features.

In [26], the authors proposed a hybrid framework that integrates wrapper-based feature selection with deep learning techniques. The framework first applies a machine learning-based wrapper feature selection method to identify the most relevant features from the UNSW-NB15 dataset. The selected feature subset is then used as input to a Multi-Head Attention-based Transformer model. Experimental results demonstrated that the proposed approach effectively reduced feature dimensionality while improving detection performance.

As summarized in Table 1, existing studies have made valuable contributions to improving intrusion detection systems through optimization-based feature selection and advanced learning models. However, these optimization-based feature selection approaches are typically implemented in a serial and centralized manner, with researchers primarily focusing on improving detection performance and reducing dimensionality while often overlooking computational cost. This limits their scalability when applied to large-scale IoT data, and the integration of such approaches with distributed computing frameworks remains largely unexplored. To address these limitations, this paper proposes DISFS-IDS using Apache Spark, which employs an independent-island architecture, a two-level partitioning strategy, and an adaptive threshold aggregation mechanism to enable scalable distributed feature selection for large-scale IoT intrusion detection systems.

Table 1. Summary of Recent Related Work on Optimization-Based Intrusion Detection Systems (2024–2025).

Ref.	Year	Technique	Scalability	Primary Focus
[10]	2024	ARO + Decision Tree	Serial	IoT botnet detection with feature reduction
[11]	2024	GWO + Deep Learning model	Serial	GWO + IDS feature selection with deep learning for improved accuracy and efficiency
[12]	2025	SFO + WOA + PSO + K-means (hybrid)	Serial	Botnet detection with feature selection and hybrid optimization
[13]	2025	Binary GWO + ESNN + AOA	Serial	Cyberattack detection in cyber-physical system with feature selection and DL optimization

Table 1. Cont.

Ref.	Year	Technique	Scalability	Primary Focus
[14]	2025	GWO + QBBA (hybrid) + machine learning classifiers	Serial	IDS feature selection and attack detection with reduced features
[17]	2025	GWO + LOA + Random Forest	Serial	IDS feature selection and intrusion classification with improved performance
[18]	2025	Filter methods + Fuzzy TOPSIS + DGWO	Serial	Multi-phase hybrid optimization for IDS feature selection
[19]	2025	Improved Chimp Optimization Algorithm (IChOA) + Autoencoder + TCN-GRU (DINet)	Serial	IDS feature selection and deep learning-based intrusion detection
[20]	2025	Statistical methods + Metaheuristic FS + LSTM + XAI (LIME, SHAP)	Serial	Explainable IDS with feature selection and improved detection accuracy
[21]	2025	PSO + ML/DL models (CatBoost, CNN, LSTM)	Serial	IoT IDS feature selection and anomaly detection with improved performance
[22]	2025	EGTO (Enhanced Gorilla Troops Optimizer)	Serial	IDS feature selection optimization for improved intrusion detection
[23]	2025	Scatter Search + SVM	Serial	IDS feature selection with reduced features and high detection accuracy
[24]	2025	LDA + EWOA + RF + XGBoost	Serial	IDS feature selection and classification with improved accuracy and reduced false alarms
[25]	2025	SHAP + Correlation FS + Information Gain + DNN	Serial	IDS feature selection using XAI for improved accuracy
[26]	2025	Wrapper FS + Transformer (Multi-Head Attention)	Serial	IDS feature selection and intrusion detection with improved accuracy
Our Proposed DISFS-IDS	–	Distributed Island-Based Feature Selection (DISFS-IDS) + Decision Tree	Distributed	Scalable distributed IDS feature selection with reduced features and high detection performance

3. Starfish Optimization Algorithm (SFOA)

In [15], Zhong et al. proposed a novel optimization method known as the Starfish Optimization Algorithm (SFOA) that simulates the exploring, preying, and regeneration behaviors of starfish. In SFOA, a hybrid exploration strategy combines five- and one-dimensional search mechanisms. This hybrid pattern addresses the slow convergence of unidimensional search and the inefficiency of vectorial search methods. During the exploration phase, the SFOA applies a five-dimensional search approach for optimization problems with dimensionality (D) greater than five and a unidimensional search for dimensionality less than or equal to five dimensions. All notations used throughout this paper are summarized in Table 2.

Table 2. Summary of notations used throughout the paper.

Notation	Description	Notation	Description
t	Current Iteration	T_{max}	Maximum iterations
d	Dimension	X_d^t	Current position of a starfish
X_d^{t+1}	Updated position of a starfish	$Best_d^t$	Current best position
E_t	Energy of the starfish	d_m	Distance from the global best
P_{size}	Population size	$Fitness$	Fitness value
$Error$	Classification error	w_1, w_2	Weighting factors
M	Number of islands	S_j	Feature importance score
$x_j^{(m)}$	Feature selection indicator	$F_{selected}$	Selected features
F_{total}	Total features	T	Adaptive threshold

In a problem with $D > 5$, the starfishes (individuals) utilize all five arms to explore the neighboring regions of the search space efficiently. Furthermore, the movement of each arm is influenced by the best starfish (*Best*) in the population, improving search efficiency. Mathematically, the position of the starfish X at iteration (t) is updated as follows:

$$X_d^{t+1} = \begin{cases} X_d^t + \alpha(Best_d^t - X_d^t) \cos \theta, & \text{if } r \leq 0.5 \\ X_d^t - \alpha(Best_d^t - X_d^t) \sin \theta, & \text{if } r > 0.5 \end{cases} \quad (1)$$

$$\alpha = (2r - 1)\pi \quad (2)$$

$$\theta = \frac{\pi}{2} \cdot \frac{t}{T_{max}} \quad (3)$$

where t is the current iteration, *Best* refers to the best starfish in the population at iteration t , r is a random number in the range $[0, 1]$, and T_{max} represents the total number of iterations. The variable d denotes the chosen dimension, while α and θ adjust the influence of the distance between the current position of a starfish and the best starfish along the selected update dimensions. To maintain search capability and enhance efficiency compared to the vectorial search pattern, only five dimensions of the position are updated. However, if an updated dimension falls outside the predefined boundary, the arm remains at its previous position.

When $D \leq 5$, the exploration phase updates the positions of the starfishes using a one-dimensional search approach. In this case, each starfish explores the search space through only one of its arms, and its movement is influenced by positional information obtained from other starfishes. The position update is given as follows:

$$X_d^{t+1} = E_t X_d^t + \alpha_1 (X_{r1,d}^t - X_d^t) + \alpha_2 (X_{r2,d}^t - X_d^t) \quad (4)$$

$$E_t = \frac{T_{max} - t}{T_{max}} \cos \theta \quad (5)$$

where X_{r1} and X_{r2} are two randomly selected starfishes, and α_1 and α_2 are two random numbers in the range $(-1, 1)$. Here, if the new position of a starfish lies outside the predefined boundary, it is more likely to remain at its previous one rather than move to the updated position.

For the exploitation phase, SFOA mimics the preying behavior and regeneration strategy of starfish. During the preying process, a parallel bidirectional search strategy is employed, utilizing information from other starfishes as well as the current best starfish in the population. In this phase, five distances between the position of the best starfish and the remaining starfishes are first calculated. Then, two of these distances are randomly selected and used to update the position of each starfish. The distances are defined as follows:

$$d_m = (Best^t - X_{m_p}^t) \quad (6)$$

Here, d_m represents the five distances calculated between the best starfish in the population and the other starfishes, while m_p denotes five starfishes randomly selected from the population. Based on this, the position update rule during the preying behavior is formulated as follows:

$$X^{t+1} = X^t + r_1 d_{m1} + r_2 d_{m2} \quad (7)$$

Here, r_1 and r_2 are random numbers in the interval $(0, 1)$, and d_{m1} and d_{m2} are randomly selected distances from the set d_m .

Moreover, SFOA models the regeneration behavior observed in starfish, which are vulnerable to predators during hunting. This mechanism is applied only to the last starfish in the population ($i = N$). Since regeneration is inherently a slow biological process, the corresponding position update is modeled using a slower movement strategy. This phase

is essential for avoiding local optima and improving global convergence. Mathematically, the update of the last starfish during the regeneration phase is defined as follows:

$$X^{t+1} = \exp(-t \times P_{size} / T_{max}) X^t \quad (8)$$

where P_{size} is the population size.

In SFOA, the G_P parameter plays an important role in guiding the search behavior by regulating the trade-off between exploration and exploitation. A higher value of G_P encourages SFOA to explore a wider region of the search space, helping to avoid premature convergence to local optima. In contrast, a lower value directs the search toward exploitation, allowing SFOA to focus more on refining the best solutions found so far.

4. DISFS-IDS: A Distributed Intrusion Detection System Using Island-Based Feature Selection

In this paper, we propose an island-based intrusion detection system (IDS) framework using Apache Spark, called DISFS-IDS. In the proposed framework, the dataset is partitioned into multiple islands while preserving the class distribution across all partitions. In addition to data partitioning, the global population is also divided among the islands, such that each island maintains and evolves its own local sub-population of candidate solutions.

Each island independently applies the Starfish Optimization Algorithm (SFOA) to its local data subset to identify an optimal feature subset for the Decision Tree (DT) model (kernel). Notably, no migration is performed between islands during the optimization process, allowing fully independent parallel exploration and reducing communication overhead.

The DT model was selected during the optimization process because it has been widely adopted in wrapper-based feature selection studies due to its efficiency and ability to provide reliable fitness evaluations while maintaining a reasonable computational cost. Nevertheless, the proposed DISFS-IDS framework is classifier-independent and can be integrated with different machine learning models.

After the completion of all local optimization processes, the selected feature subsets from all islands are collected and integrated using a rank-based aggregation mechanism to construct a reliable and high-quality global feature subset. Figure 1 illustrates the architecture of the DISFS-IDS.

In the optimization phase, SFOA is employed to identify an optimal subset of features for the DT classifier to achieve high detection performance. The individuals (starfishes) are encoded as a binary string, as shown in Figure 2.

Each individual represents a potential solution (subset of features) for the problem. The fitness function, which the SFOA aims to minimize, is computed as a combination of classification error and the proportion of selected features. The fitness function [27] is presented as

$$\text{Fitness} = w_1 \cdot \text{Error} + w_2 \cdot \frac{F_{\text{selected}}}{F_{\text{total}}} \quad (9)$$

Here, Error refers to the classification error rate of the DT model evaluated on the validation data, while F_{selected} and F_{total} represent the number of selected and total features, respectively. The coefficients w_1 and w_2 are weighting factors that balance the classification error and the selected feature ratio, with their sum equal to one ($w_1 + w_2 = 1$). This balance ensures that the DISFS-IDS not only finds feature subsets that achieve high classification performance but also promotes feature reduction. In this work, the weights were set to $w_1 = 0.99$ and $w_2 = 0.01$ [27].

For selecting the final feature subset, an adaptive threshold-based strategy is used to determine the final feature subset based on the aggregated feature importance scores. First, the importance score of each feature is computed by aggregating the feature selections

obtained from all islands. Let M denote the number of islands. The importance score of feature j is defined as

$$S_j = \frac{1}{M} \sum_{m=1}^M x_j^{(m)} \quad (10)$$

where $x_j^{(m)} \in \{0, 1\}$ indicates whether feature j is selected by the m -th island.

The obtained feature scores are normalized. Then, an adaptive threshold T is computed based on the mean $\mu_{\hat{S}}$ and standard deviation $\sigma_{\hat{S}}$ of the normalized scores $\hat{S}$, as follows:

$$T = \mu_{\hat{S}} + 0.5 \cdot \sigma_{\hat{S}} \quad (11)$$

Finally, features with normalized scores greater than or equal to T are selected to form the final feature subset.

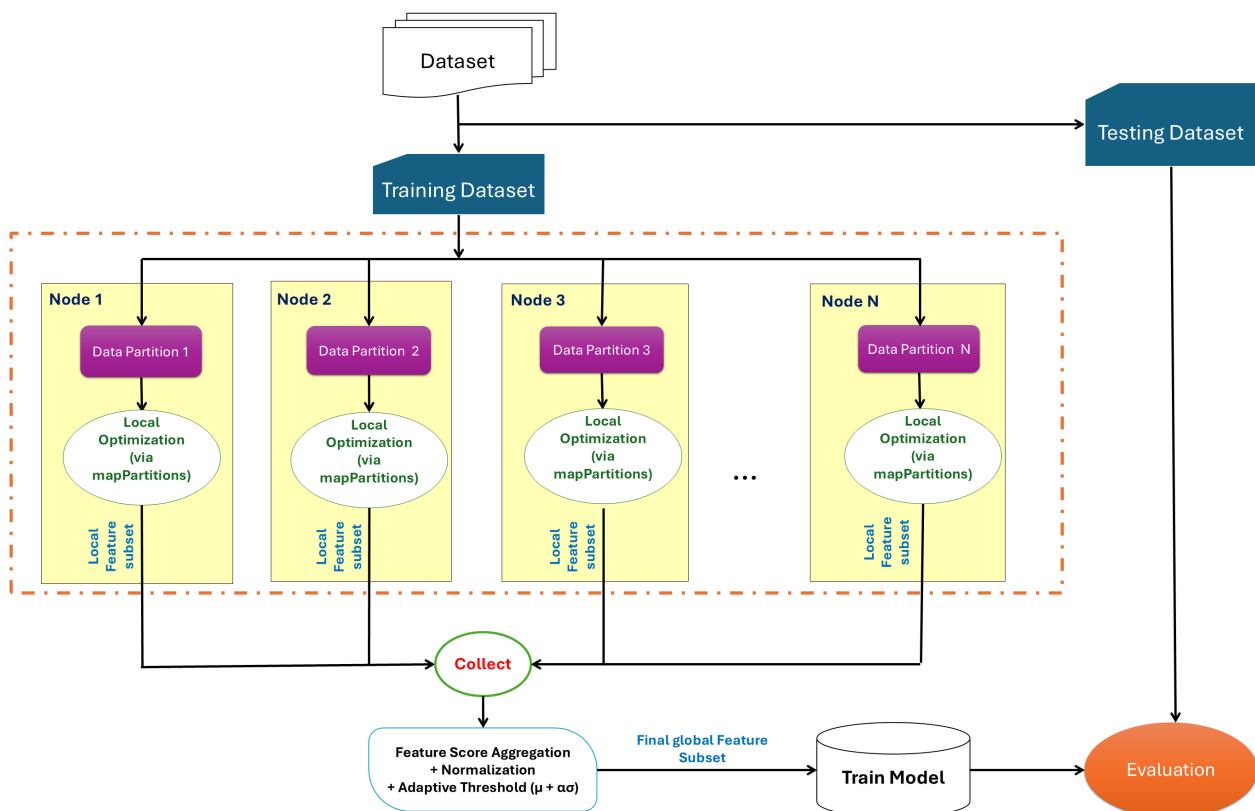


Figure 1. Architecture of the proposed DISFS-IDS framework, including data partitioning, distributed feature selection, and intrusion detection evaluation.



Figure 2. Binary encoding of an individual solution for feature selection.

5. Experimental Setup and Datasets

This section first introduces the IoT datasets used in the study and then outlines the experimental setup and evaluation metrics.

5.1. IoT Datasets

In this work, the proposed detection model was evaluated using three publicly and widely used IoT datasets:

- The TON_IoT [28–35], dataset was created and generated using a testbed environment comprising IoT and IIoT devices distributed across edge, fog, and cloud layers. It contains normal network traffic, and various cyberattacks—such as DoS, DDoS, and ransomware—were executed against web applications, IoT gateways, and computer systems within the IIoT network. This dataset contains 21,104 instances and 124 features.
- The CIC-MalMem-2022 [36] dataset is a publicly available dataset that was generated by the Canadian Institute for Cybersecurity. This dataset is well-balanced and contains a total of 58,596 memory dump records, equally divided between benign and malicious samples. It includes various malware types such as spyware, ransomware, and trojan horses. Each sample is represented by 57 features that capture memory-based indicators of malicious behavior.
- The CIC-Darknet [37] dataset was generated to represent Darknet traffic, including VPN and Tor traffic, and was originally introduced by the Canadian Institute for Cybersecurity. This dataset contains over 140,000 instances with 85 features, derived from both normal and malicious activities.

The IoT datasets were selected to provide a comprehensive evaluation of the performance of DISFS-IDS across different IoT attack scenarios. In addition, their high dimensionality and large number of instances create a challenging environment for assessment.

The studied IoT datasets were preprocessed consistently for DISFS-IDS. Identifier features such as IP addresses were removed, instances with missing values were excluded, and all remaining features were normalized to [0, 1].

5.2. Setup and Metrics

In this paper, the experiments were conducted on an Amazon EMR cluster using m4.xlarge instances to evaluate the performance and scalability of DISFS-IDS. Each m4.xlarge instance contains four virtual CPUs and 16 GB of memory.

To evaluate the robustness and effectiveness of the proposed DISFS-IDS across all datasets, several performance metrics were considered, including accuracy, F1-score, and feature reduction ratio.

For scalability analysis, the speedup metric was used to measure the efficiency of parallel execution as the number of computing nodes increases, while keeping the problem size constant (i.e., dataset size and population size). The speedup is defined as:

$$\text{Speedup}(p) = \frac{T_{\text{serial}}}{T_p} \quad (12)$$

where T_{serial} denotes the execution time of the serial implementation, T_p represents the execution time using p computing nodes (or islands), and p is the number of nodes.

6. Experiments and Results

This section presents a comprehensive evaluation of DISFS-IDS. First, a detailed analysis of its detection performance across the studied IoT datasets is provided. Next, the scalability performance of DISFS-IDS is evaluated in terms of speedup. Then, a computational complexity analysis is presented. Finally, an ablation study is conducted. In all experiments, the population size and the maximum number of iterations were fixed at 200 and 100, respectively.

6.1. Detection Performance Analysis

This subsection evaluates the detection performance of DISFS-IDS through comparisons with the serial implementation, distributed feature selection baselines, alternative

optimization algorithms, and state-of-the-art methods. In addition, the generalization capability of the selected feature subsets is investigated.

6.1.1. Comparison with the Serial Baseline

The performance and effectiveness of the proposed DISFS-IDS model were evaluated on both binary and multiclass IoT datasets. The DISFS-IDS was executed using 2 and 4 islands (worker nodes) on the Amazon EMR cluster, referred to as DISFS-IDS (I2) and DISFS-IDS (I4), respectively. The evaluation results are reported in terms of accuracy and F1-score. Furthermore, the obtained results were compared with those of the serial version to highlight the advantages of the distributed framework.

For a fair comparison with the serial method, both the serial method and DISFS-IDS use the same population size of 200 and a maximum of 100 iterations. These values were selected to provide a balance between search capability and computational cost and were kept constant across all experiments. In addition, the G_P parameter of SFOA is set to 0.5 following the recommendation of the original SFOA paper [15], where this value was determined through sensitivity analysis. The DT classifier is used as the base classifier with identical parameter settings in all experiments.

To ensure statistical reliability, each experiment was repeated 20 times using different random train–test splits. The mean and 95% confidence interval margin of error (CI95_MOE) for each metric were reported. In addition, the Wilcoxon signed-rank test was performed at a significance level of 0.05, and the corresponding p-values were provided.

Table 3 presents the experimental results obtained by DISFS-IDS and the serial feature selection approach in terms of accuracy, F1-score, and the number of selected features. As shown in the table, both DISFS-IDS (I2) and DISFS-IDS (I4) achieved a detection performance of 99.99% on the TON_IoT binary dataset, which is very close to that of the serial method. The Wilcoxon signed-rank test indicates that there is no statistically significant difference in accuracy or F1-score between the serial method and DISFS-IDS using either 2 or 4 islands. However, DISFS-IDS (I2) and DISFS-IDS (I4) selected fewer features, approximately 39–40, compared to the serial method (approximately 46 features).

Table 3. Performance comparison of Serial SFOA and DISFS-IDS across the studied datasets.

Dataset	Method	Accuracy (%)	F1-Score (%)	# Selected Features
TON_IoT	Serial (S)	100.00 ± 0.00	100.00 ± 0.00	46.7 ± 1.75
	DISFS-IDS (I2)	99.99 ± 0.01	99.99 ± 0.01	39.5 ± 1.51
	DISFS-IDS (I4)	99.99 ± 0.005	99.99 ± 0.005	40.2 ± 1.87
	p-value (S vs. I2)	0.5000 (T)	0.5000 (T)	0.0019 (W)
	p-value (S vs. I4)	1.0000 (T)	1.0000 (T)	0.0019 (W)
CIC-MalMem-2022	Serial (S)	99.95 ± 0.015	99.95 ± 0.015	18.73 ± 1.41
	DISFS-IDS (I2)	99.93 ± 0.031	99.93 ± 0.031	17.66 ± 1.18
	DISFS-IDS (I4)	99.94 ± 0.012	99.94 ± 0.012	16.47 ± 1.08
	p-value (S vs. I2)	0.3281 (T)	0.3281 (T)	0.3998 (T)
	p-value (S vs. I4)	0.3865 (T)	0.3865 (T)	0.0088 (W)
CIC-Darknet	Serial (S)	92.98 ± 0.128	92.92 ± 0.233	40.80 ± 2.67
	DISFS-IDS (I2)	92.76 ± 0.192	92.84 ± 0.230	23.60 ± 1.12
	DISFS-IDS (I4)	93.00 ± 0.046	92.94 ± 0.074	22.60 ± 0.60
	p-value (S vs. I2)	0.2246 (T)	0.9218 (T)	0.0019 (W)
	p-value (S vs. I4)	0.7578 (T)	0.3125 (T)	0.0019 (W)

Note: Performance values are reported as mean ± CI95_MOE. (W) indicates that DISFS-IDS performs significantly better than Serial, (T) indicates no statistically significant difference, and (L) indicates that DISFS-IDS performs significantly worse than Serial. # indicates the number of selected features.

Similar observations can be made for the CIC-MalMem-2022 binary dataset. Both versions of DISFS-IDS achieved nearly identical accuracy and F1-score values compared to the serial method, reaching approximately 99.93–99.94%, with no statistically significant difference (p -values > 0.05). In terms of the number of selected features, DISFS-IDS (I2) and DISFS-IDS (I4) reduced the feature subset to approximately 16–17 features, compared to 18 features in the serial method. Notably, DISFS-IDS (I4) showed a statistically significant reduction in the number of selected features compared to the serial approach (p -value = 0.0088).

For the CIC-Darknet multiclass dataset, a similar pattern is observed. DISFS-IDS (I2) and DISFS-IDS (I4) achieve detection performance comparable to the serial method, with no statistically significant difference in terms of accuracy and F1-score (p -values exceed 0.05). In contrast, the use of 2 and 4 islands significantly reduces the feature subset size to approximately 22–23 features, compared to around 40 features in the serial method.

Figure 3 shows the feature reduction ratio obtained by the serial method and DISFS-IDS across different datasets. As can be observed, DISFS-IDS achieves higher feature reduction ratios than the serial method across all datasets. This confirms the ability of DISFS-IDS to select a more compact feature subset while improving computational efficiency.

Overall, DISFS-IDS achieves detection performance that is statistically comparable to the serial method. In addition, it significantly reduces the number of selected features, indicating its ability to identify a more compact and informative feature subset without compromising detection performance.

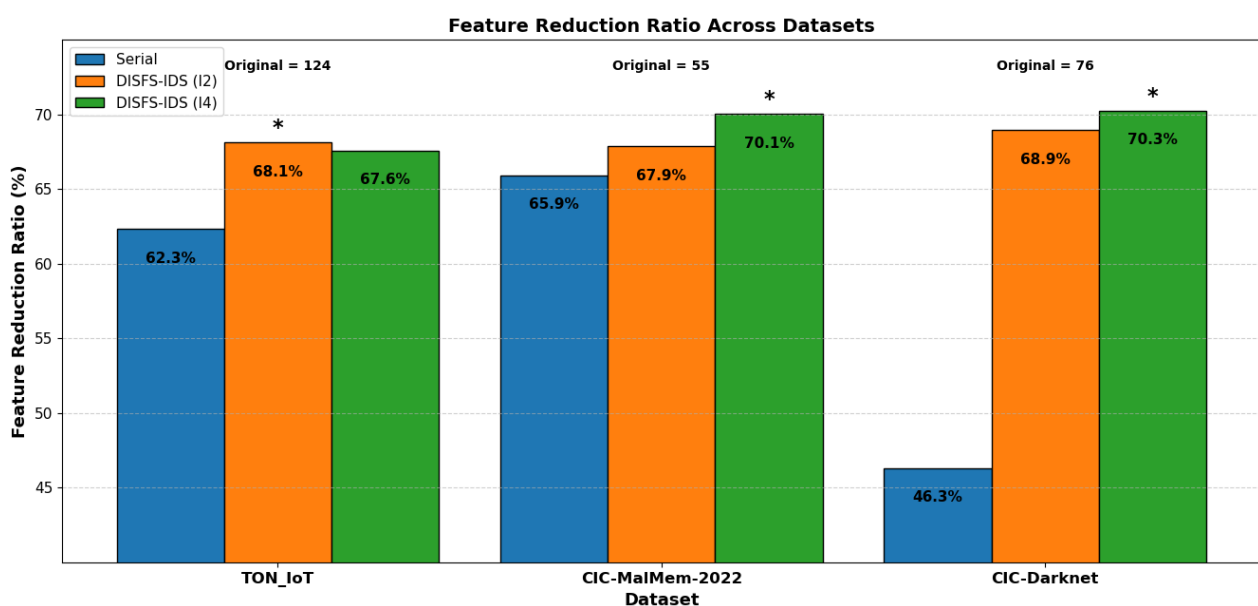


Figure 3. Feature reduction ratio (%) achieved by the serial and distributed DISFS-IDS variants using 2 and 4 islands across the TON_IoT, CIC-MalMem-2022, and CIC-Darknet datasets. The best results are marked with an asterisk (*).

6.1.2. Comparison with Distributed Feature Selection Baselines

In this experiment, we compared the performance of DISFS-IDS with three distributed feature selection baselines available in Apache Spark MLlib [38]: Spark Random Forest Feature Importance (Spark-RF), Spark Univariate Feature Selector (Spark-UFS), and Spark Variance Threshold Selector (Spark-VTS). For a fair comparison, the feature subsets selected by all Spark-based baselines were evaluated using the same Decision Tree classifier employed by DISFS-IDS. Furthermore, each Spark-based baseline was independently executed 20 times using different random train–test splits, and the reported results correspond to the average performance over all runs.

Table 4 presents the results of the distributed feature selection baselines and DISFS-IDS across the studied datasets. The results show that DISFS-IDS achieves competitive or slightly superior detection accuracy compared with the Spark-based baselines. Although Spark-RF selects fewer features on the binary datasets, DISFS-IDS maintains comparable accuracy on TON_IoT and achieves the highest accuracy on CIC-MalMem-2022. For CIC-Darknet, a multiclass dataset, DISFS-IDS outperforms the Spark-based baselines in terms of both detection accuracy and feature reduction. These findings confirm the effectiveness of DISFS-IDS in identifying informative feature subsets while maintaining competitive or superior intrusion detection performance compared with the distributed Spark-based feature selection baselines.

Table 4. Comparison of DISFS-IDS and distributed Spark-based feature selection baselines.

Dataset	Method	Accuracy (%)	# Selected Features
TON_IoT	Spark-RF	99.99 ± 0.007	19.6 ± 1.48
	Spark-UFS	99.99 ± 0.007	104.0 ± 0.00
	Spark-VTS	99.99 ± 0.007	104.7 ± 0.35
	DISFS-IDS (I2)	99.99 ± 0.010	39.5 ± 0.51
	DISFS-IDS (I4)	99.99 ± 0.005	40.2 ± 1.87
CIC-MalMem-2022	Spark-RF	99.92 ± 0.040	13.0 ± 0.65
	Spark-UFS	99.93 ± 0.030	51.1 ± 0.23
	Spark-VTS	99.93 ± 0.030	39.0 ± 0.00
	DISFS-IDS (I2)	99.93 ± 0.031	17.66 ± 1.18
	DISFS-IDS (I4)	99.94 ± 0.012	16.47 ± 1.08
CIC-Darknet	Spark-RF	92.53 ± 0.27	28.20 ± 1.54
	Spark-UFS	92.57 ± 0.25	60.90 ± 0.23
	Spark-VTS	92.57 ± 0.25	61.00 ± 0.00
	DISFS-IDS (I2)	92.76 ± 0.192	23.6 ± 1.12
	DISFS-IDS (I4)	93.00 ± 0.046	22.6 ± 0.60

Note: The best results are reported in bold. # indicates the number of selected features.

6.1.3. Comparison with Alternative Optimization Algorithms

In this subsection, we evaluate the performance of SFOA by comparing it with other well-known metaheuristic algorithms, including Particle Swarm Optimization (PSO) [39] and Grey Wolf Optimization (GWO) [40], as wrapper-based feature selection methods under the same distributed framework. Since DISFS-IDS is designed as an optimizer-independent framework, different optimization algorithms can be integrated into the framework without modifying the underlying distributed architecture. To ensure a fair comparison, each algorithm was independently executed 20 times using identical parameter settings, including four islands and the same Decision Tree (DT) classifier. The comparison aims to assess the effectiveness of SFOA relative to alternative optimization algorithms in terms of detection performance and feature reduction capability across the evaluated datasets.

Table 5 presents the results obtained by SFOA, PSO, and GWO under the proposed DISFS-IDS framework. For the TON_IoT dataset, all optimizers achieved nearly identical detection performance, with an accuracy of 99.99%. However, GWO selected the smallest feature subset compared to PSO and SFOA. For the CIC-MalMem-2022 dataset, SFOA and PSO achieved the highest accuracy (99.94%), while SFOA selected the fewest features. For the multiclass CIC-Darknet dataset, SFOA outperformed both PSO and GWO in terms of accuracy, achieving 93.00%, while also selecting fewer features. These findings demonstrate that SFOA provides the most consistent balance between detection performance and feature reduction across the evaluated datasets, supporting its selection as the optimization engine of the proposed DISFS-IDS framework.

Table 5. Comparison of SFOA, PSO, and GWO under the DISFS-IDS framework. Results are reported as Mean $\pm$ CI95_MOE over 20 independent runs.

Dataset	Optimizer	Accuracy (%)	# Selected Features
TON_IoT	SFOA	99.99 $\pm$ 0.005	40.2 $\pm$ 1.87
	PSO	99.99 $\pm$ 0.005	39.4 $\pm$ 1.22
	GWO	99.99 $\pm$ 0.005	38.9 $\pm$ 1.18
CIC-MalMem-2022	SFOA	99.94 $\pm$ 0.012	16.47 $\pm$ 1.08
	PSO	99.94 $\pm$ 0.016	17.27 $\pm$ 0.77
	GWO	99.92 $\pm$ 0.026	18.20 $\pm$ 1.45
CIC-Darknet	SFOA	93.00 $\pm$ 0.046	22.6 $\pm$ 0.60
	PSO	92.09 $\pm$ 0.66	23.4 $\pm$ 2.26
	GWO	91.96 $\pm$ 1.41	23.4 $\pm$ 1.11

Note: The best results are reported in bold. # indicates the number of selected features.

6.1.4. Generalization Analysis

To investigate the classifier dependency of wrapper-based feature selection, we evaluate the generalization capability of the feature subsets selected by DISFS-IDS using a Decision Tree (DT) as the wrapper classifier. The selected feature subsets were subsequently evaluated using Random Forest (RF) and XGBoost (XGB) classifiers. These classifiers represent different learning paradigms, namely bagging and boosting, respectively. In this experiment, DISFS-IDS was executed using four islands, and the results are presented in Table 6.

Table 6. Performance comparison of different classifiers using the features selected by DISFS-IDS (I4).

Dataset	Classifier	Accuracy (%) $\pm$ CI95_MOE	F1-Score (%) $\pm$ CI95_MOE
TON_IoT	DT	99.99 $\pm$ 0.01	99.99 $\pm$ 0.01
	RF	99.99 $\pm$ 0.01	99.99 $\pm$ 0.01
	XGB	99.95 $\pm$ 0.04	99.95 $\pm$ 0.04
CIC-MalMem-2022	DT	99.94 $\pm$ 0.012	99.94 $\pm$ 0.012
	RF	99.99 $\pm$ 0.040	99.99 $\pm$ 0.040
	XGB	100.00 $\pm$ 0.050	100.00 $\pm$ 0.050
CIC-Darknet	DT	93.00 $\pm$ 0.046	92.94 $\pm$ 0.074
	RF	97.00 $\pm$ 0.38	97.00 $\pm$ 0.38
	XGB	97.61 $\pm$ 0.12	97.61 $\pm$ 0.11

Note: The best results are reported in bold.

As shown in Table 6, DT, RF, and XGB classifiers achieved outstanding detection performance on the TON_IoT and CIC-MalMem-2022 datasets. For the multiclass CIC-Darknet dataset, RF and XGB significantly outperformed DT, achieving accuracy values of 97.00% and 97.59%, respectively, compared to 93.00% for DT. These results confirm that the selected feature subsets are not specific to the Decision Tree classifier used during wrapper optimization. In addition, the findings indicate that the feature subsets selected by DISFS-IDS generalize well across different classifiers, demonstrating the robustness and effectiveness of the DISFS-IDS.

6.1.5. Comparison with State-of-the-Art Methods

To demonstrate the effectiveness of DISFS-IDS, its detection performance using Decision Tree (DT), Random Forest (RF), and XGBoost (XGB) classifiers was compared with recent related studies, as shown in Table 7. The selected studies were chosen based on their reported performance on the evaluated datasets. The results indicate that DISFS-IDS achieves competitive or superior detection performance across all datasets. Furthermore, DISFS-IDS maintains high detection accuracy while reducing the feature dimensionality

through a distributed feature selection process, demonstrating the effectiveness of the proposed distributed framework for scalable IoT intrusion detection.

Table 7. Comparison of the classification accuracy achieved by DISFS-IDS and recent related studies.

Dataset	Reference	Method	Accuracy (%)
TON_IoT	Mohamed et al. [41]	ReliefF + MNN	98.39
	Yaras et al. [42]	CNN-LSTM	98.75
	This Work	DISFS-IDS (DT)	99.99
		DISFS-IDS (RF)	99.99
		DISFS-IDS (XGB)	99.95
CIC-MalMem-2022	Carrier et al. [36]	Ensemble	99.00
	Dener et al. [43]	DCNN	99.97
	Roy et al. [44]	MalHyStack	99.98
	This Work	DISFS-IDS (DT)	99.94
		DISFS-IDS (RF)	99.99
		DISFS-IDS (XGB)	100.00
CIC-Darknet	Karagöl et al. [45]	RF + SMOTE	97.16
	Akour et al. [46]	RF + PCA	97.60
	This Work	DISFS-IDS (DT)	93.00
		DISFS-IDS (RF)	97.00
		DISFS-IDS (XGB)	97.61

Note: The best results are reported in bold.

6.2. Scalability Analysis

In this experiment, we evaluate the scalability performance of DISFS-IDS compared to the serial method. Table 8 presents the runtime (in seconds) for both the serial method and DISFS-IDS, along with the corresponding speedup of DISFS-IDS across all datasets.

For the TON_IoT dataset, the execution time decreases from 4709 s in the serial version to 1058 s and 412 s using 2 and 4 islands, respectively, corresponding to speedup values of 4.45× and 11.43×.

Table 8. Execution Time and Speedup Analysis of DISFS-IDS Compared with the Serial Baseline.

Dataset	Execution Time (s)			Speedup	
	Serial (S)	DISFS-IDS (I2)	DISFS-IDS (I4)	I2	I4
TON_IoT	4709	1058	412	4.45	11.43
CIC-MalMem-2022	6669	947	362	7.04	18.32
CIC-Darknet	29,525	4802	1935	6.15	15.26

For the CIC-MalMem-2022 dataset, DISFS-IDS significantly reduced the execution time from 6669 s to 947 s (I2) and 362 s (I4), achieving speedups of 7.04× and 18.32×, respectively.

For the CIC-Darknet dataset, which has the highest computational cost due to its larger size, DISFS-IDS reduced the execution time from 29,525 s in the serial version to 4802 s (I2) and 1935 s (I4), resulting in speedups of 6.15× and 15.26×, respectively.

Another notable observation from the scalability results is the superlinear speedup achieved with 2 and 4 computing nodes. This is due to the two-level partitioning strategy of DISFS-IDS, where both the dataset and the population are distributed across islands. Thus, each island processes a smaller workload, reducing the cost of classifier-based fitness evaluation. In addition, smaller data partitions improve memory utilization and cache

efficiency, while the independent island architecture minimizes communication overhead. Overall, the scalability results demonstrate that DISFS-IDS achieves high speedup as the number of islands increases while maintaining high parallel efficiency. Furthermore, it scales well with an increasing number of islands. A detailed computational complexity analysis is presented in the next section.

Although the scalability analysis was conducted using 2 and 4 islands (computing nodes), the DISFS-IDS framework is designed to support a larger number of islands. As the number of islands increases, the computational workload assigned to each island decreases, potentially leading to further reductions in execution time. However, at higher levels of parallelism, factors such as Spark scheduling overhead, resource contention, and communication overhead may become more significant. Nevertheless, DISFS-IDS employs an independent island architecture that minimizes synchronization and communication overhead, enabling the framework to maintain good scalability on larger clusters. Furthermore, Apache Spark provides distributed memory management and fault-tolerance mechanisms that enhance resource utilization and ensure reliable execution in distributed environments.

6.3. Computational Complexity Analysis

Let N be the number of instances, D the number of features, P_{size} the population size, T the number of iterations, and K the number of islands. Let DT_{time} denote the computational cost of training and evaluating the Decision Tree classifier for a given feature subset.

In the serial version, the computational complexity is dominated by the fitness evaluation, which requires training a Decision Tree for each candidate solution. Thus, the overall complexity can be expressed as:

$$O(P_{size} \times T \times DT_{time})$$

In the proposed distributed framework, both the dataset and the population are divided across K islands. Each island processes approximately $\frac{N}{K}$ instances and $\frac{P_{size}}{K}$ candidate solutions. Accordingly, the computational complexity per island becomes:

$$O\left(\frac{P_{size}}{K} \times T \times DT_{time}^{(K)}\right)$$

where $DT_{time}^{(K)}$ denotes the Decision Tree training and evaluation cost on $\frac{N}{K}$ instances.

This analysis demonstrates that the proposed distributed framework significantly reduces the computational cost compared to the serial version by leveraging parallel execution and distributing the workload across islands. Furthermore, communication and synchronization overhead are minimized through the use of an independent island architecture. However, the actual performance in distributed environments may also be affected by factors such as Spark scheduling overhead, data serialization, and memory constraints.

6.4. Ablation Study

In this subsection, an ablation study is conducted to investigate the impact of the adaptive thresholding strategy and the migration-free island mechanism in DISFS-IDS. All experiments were performed using four islands on the CIC-Darknet multiclass dataset to evaluate the contribution of each component to the overall framework performance.

6.4.1. Effect of the Thresholding Strategy

To assess the impact of the thresholding mechanism used in DISFS-IDS, an ablation study was conducted by comparing the mean thresholding strategy ($T = \mu_S$) with the proposed mean-standard deviation thresholding strategy ($T = \mu_S + 0.5 \cdot \sigma_S$). In this experiment, DISFS-IDS was configured with four islands and SFOA as the underlying

optimization algorithm. Each strategy was run 20 independent times using different random train–test splits.

Table 9 presents the results obtained using the mean threshold and the proposed mean–standard deviation thresholding strategy. For the TON_IoT and CIC-MalMem-2022 datasets, both strategies achieved the same detection accuracy, while the proposed mean–standard deviation threshold selected fewer features. For the multiclass CIC-Darknet dataset, the proposed strategy improved the detection accuracy to 93.00% compared with the mean threshold, while also selecting fewer features. These findings demonstrate the effectiveness of the proposed thresholding strategy in identifying more informative feature subsets without reducing detection performance.

Table 9. Ablation study of the thresholding strategy. Results are reported as Mean $\pm$ CI95_MOE over 20 runs.

Dataset	Threshold Strategy	Accuracy (%)	# Selected Features
TON_IoT	$T = \mu_S$	99.99 ± 0.011	65.80 ± 3.87
	$T = \mu_S + 0.5 \cdot \sigma_S$	99.99 ± 0.005	40.20 ± 1.87
CIC-MalMem-2022	$T = \mu_S$	99.94 ± 0.053	26.60 ± 2.72
	$T = \mu_S + 0.5 \cdot \sigma_S$	99.94 ± 0.012	16.47 ± 1.08
CIC-Darknet	$T = \mu_S$	92.60 ± 0.709	36.60 ± 2.58
	$T = \mu_S + 0.5 \cdot \sigma_S$	93.00 ± 0.046	22.60 ± 0.60

Note: The best results are reported in bold. # indicates the number of selected features.

6.4.2. Comparison of Migration and Independent Islands

To investigate the impact of the independent-island architecture in DISFS-IDS on detection performance, feature reduction, and computational efficiency, an ablation study was conducted by comparing two variants of the framework: a migration-based variant and an independent-island variant. In the migration-based variant, the best solutions are periodically exchanged among islands at fixed migration intervals. In contrast, the independent-island variant does not perform migration, allowing islands to evolve independently throughout the optimization process. This design helps preserve population diversity while reducing communication and synchronization overhead in Apache Spark.

In this experiment, both variants of DISFS-IDS were executed using four islands, with SFOA as the underlying optimization algorithm. The evaluation was conducted on two datasets: CIC-MalMem-2022, representing a binary classification problem, and CIC-Darknet, representing a multiclass classification problem. Each variant was executed 20 times using different random train–test splits. Table 10 presents the results obtained by the migration-based and independent-island variants.

Table 10. Comparison of migration and independent islands in DISFS-IDS.

Dataset	Method	Accuracy (%)	# Selected Features	Execution Time (s)
CIC-MalMem-2022	Independent Islands	99.94 ± 0.012	16.47 ± 1.08	362 ± 5
	Migration-based	99.94 ± 0.016	16.60 ± 0.86	409 ± 10
CIC-Darknet	Independent Islands	93.00 ± 0.046	22.60 ± 0.60	1935 ± 15
	Migration-based	92.93 ± 0.51	23.20 ± 2.38	2045 ± 12

Note: # indicates the number of selected features.

As shown in Table 10, the migration-based and independent-island variants achieved nearly identical performance in terms of detection performance and feature reduction on both the CIC-MalMem-2022 and CIC-Darknet datasets. However, the independent-island variant achieved lower execution times across both datasets than the migration-based variant. This improvement can be attributed to the elimination of the communication and

synchronization overhead associated with exchanging solutions among islands. These results demonstrate that the independent-island architecture achieves a balance between detection performance and computational efficiency, supporting the use of the independent-island architecture in DISFS-IDS.

7. Conclusions

In this paper, we proposed DISFS-IDS, a scalable island-based feature selection framework for IoT intrusion detection using Apache Spark. DISFS-IDS adopts a two-level partitioning strategy to distribute the workload across worker nodes. The primary objective of the framework is to reduce computational time and dataset dimensionality while maintaining high detection performance.

Comprehensive experiments were conducted using three widely used IoT datasets covering both binary and multiclass classification scenarios. The experimental results demonstrated that DISFS-IDS achieves statistically comparable detection performance, in terms of accuracy and F1-score, to the serial implementation. Furthermore, DISFS-IDS achieves competitive or superior detection performance while maintaining effective feature reduction compared with Spark-based filter feature selection methods. Additional analyses showed that the proposed framework scales efficiently with increasing dataset size and dimensionality, making it suitable for large-scale IoT intrusion detection environments.

Overall, the results confirm that DISFS-IDS effectively balances detection performance, feature reduction, and computational efficiency, highlighting its potential as a practical distributed feature selection framework. Although DISFS-IDS achieved promising results, its effectiveness has only been evaluated on widely used IoT intrusion detection datasets. Therefore, its applicability to other cybersecurity datasets and real-world streaming environments remains to be investigated. Furthermore, large-scale deployment factors, such as network latency and fault tolerance, were not extensively evaluated and remain subjects for future investigation.

Future work will investigate alternative strategies for constructing the final feature subset, such as weighted majority voting. In addition, we plan to evaluate the proposed framework in other application domains, including healthcare, and conduct more comprehensive scalability and latency analyses using real-world large-scale datasets.

Author Contributions: Conceptualization, J.A.-S. and A.A.M.; methodology, J.A.-S. and A.A.M.; software, J.A.-S. and A.A.M.; validation, J.A.-S. and A.A.M.; writing—review and editing, J.A.-S. and A.A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used in this study are publicly available and have been properly referenced in the manuscript. The TON-IoT dataset is available from the UNSW Canberra Cyber repository (<https://research.unsw.edu.au/projects/toniot-datasets>, accessed on 1 March 2026), while the CIC-MalMem-2022 dataset (<https://www.unb.ca/cic/datasets/malmem-2022.html>, accessed on 1 March 2026) and the CIC-Darknet dataset (<https://www.unb.ca/cic/datasets/darknet2020.html>, accessed on 1 March 2026) are available from the Canadian Institute for Cybersecurity (CIC).

Acknowledgments: Our research reflects the collaborative efforts of several institutions whose contributions were essential to the study's success. We extend our gratitude to Prince Sultan University in KSA and Tafila Technical University.

Conflicts of Interest: The authors declare that they have no conflicts of interest.

References

1. Al-Kasassbeh, M.; Almseidin, M.; Alrfou, K.; Kovacs, S. Detection of IoT-botnet attacks using fuzzy rule interpolation. *J. Intell. Fuzzy Syst.* **2020**, *39*, 421–431. [\[CrossRef\]](#)
2. Makris, I.; Karampasi, A.; Radoglou-Grammatikis, P.; Episkopos, N.; Iturbe, E.; Rios, E.; Piperigkos, N.; Lalos, A.; Xenakis, C.; Lagkas, T.; et al. A comprehensive survey of Federated Intrusion Detection Systems: Techniques, challenges and solutions. *Comput. Sci. Rev.* **2025**, *56*, 100717. [\[CrossRef\]](#)
3. Buyuktanir, B.; Altinkaya, Ş.; Karatas Baydogmus, G.; Yildiz, K. Federated learning in intrusion detection: Advancements, applications, and future directions. *Clust. Comput.* **2025**, *28*, 473. [\[CrossRef\]](#)
4. Mamdouh, H.; Tarek, M.; Radwan, A.; Saeed, A.; Abdeen, A.; Ashraf, M.; Fouad, K.M.; Abdelbaky, I. Apache Spark Powered: Enhancing Network Intrusion Detection System Using Random Forest. In *Proceedings of the 2024 6th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*; IEEE: Piscataway, NJ, USA, 2024; pp. 289–294.
5. Hagar, A.A.; Gawali, B.W. Apache spark and deep learning models for high-performance network intrusion detection using CSE-CIC-IDS2018. *Comput. Intell. Neurosci.* **2022**, *2022*, 3131153. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Khurma, R.A.; Aljarah, I.; Sharieh, A.; Mirjalili, S. Evolopy-fs: An open-source nature-inspired optimization framework in python for feature selection. In *Evolutionary Machine Learning Techniques: Algorithms and Applications*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 131–173.
7. Azar, A.; Khan, Z.; Amin, S.; Fouad, K. Hybrid global optimization algorithm for feature selection. *Comput. Mater. Contin.* **2023**, *74*, 2021. [\[CrossRef\]](#)
8. Faris, H.; Mafarja, M.M.; Heidari, A.A.; Aljarah, I.; Al-Zoubi, A.; Mirjalili, S.; Fujita, H. An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowl.-Based Syst.* **2018**, *154*, 43–67. [\[CrossRef\]](#)
9. Abedi, F.; Ghanimi, H.M.; Algarni, A.D.; Soliman, N.F.; El-Shafai, W.; Abbas, A.H.; Kareem, Z.H.; Hariz, H.M.; Alkhayyat, A. Chimp Optimization Algorithm Based Feature Selection with Machine Learning for Medical Data Classification. *Comput. Syst. Sci. Eng.* **2023**, *47*, 2791–2814. [\[CrossRef\]](#)
10. Almseidin, M.; Al-Sawwa, J.; Alkasassbeh, M.; Alzubi, M.; Alrfou, K. DT-ARO: Decision tree-based artificial rabbits optimization to mitigate IoT botnet exploitation. *J. Netw. Syst. Manag.* **2024**, *32*, 14.
11. Elsaid, S.A.; Shehab, E.; Mattar, A.M.; Azar, A.T.; Hameed, I.A. Hybrid intrusion detection models based on GWO optimized deep learning. *Discov. Appl. Sci.* **2024**, *6*, 531. [\[CrossRef\]](#)
12. Maazalahi, M.; Hosseini, S. Machine learning and metaheuristic optimization algorithms for feature selection and botnet attack detection. *Knowl. Inf. Syst.* **2025**, *67*, 3549–3597. [\[CrossRef\]](#)
13. Al Mazroa, A.; Albogamy, F.R.; Ishak, M.K.; Mostafa, S.M. Boosting Cyberattack Detection Using Binary Metaheuristics with Deep Learning on Cyber-Physical System Environment. *IEEE Access* **2025**, *13*, 11280–11294. [\[CrossRef\]](#)
14. Alotaibi, M.; Mengash, H.A.; Alqahtani, H.; Al-Sharafi, A.M.; Yahya, A.E.; Alotaibi, S.R.; Khadidos, A.O.; Yafaz, A. Hybrid GWQBBA model for optimized classification of attacks in Intrusion Detection System. *Alex. Eng. J.* **2025**, *116*, 9–19. [\[CrossRef\]](#)
15. Zhong, C.; Li, G.; Meng, Z.; Li, H.; Yildiz, A.R.; Mirjalili, S. Starfish optimization algorithm (SFOA): A bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers. *Neural Comput. Appl.* **2025**, *37*, 3641–3683.
16. Alweshah, M.; Alhebaishan, G.A.; Kassaymeh, S.; Alkhalaileh, S.; Ababneh, M. Improving intrusion detection in the IoT with African vultures optimisation algorithm-based feature selection. *Int. J. Data Min. Model. Manag.* **2024**, *16*, 293–325. [\[CrossRef\]](#)
17. Kumar, A.; Kumar, S. Metaheuristic-Enhanced Feature Selection for High-Accuracy Intrusion Detection in Cloud Computing. *Procedia Comput. Sci.* **2025**, *259*, 640–649. [\[CrossRef\]](#)
18. Pourardebil Khah, Y.; Hosseini Shirvani, M.; Motameni, H. A hybrid machine learning approach for feature selection in designing intrusion detection systems (IDS) model for distributed computing networks. *J. Supercomput.* **2025**, *81*, 254. [\[CrossRef\]](#)
19. Rani, Y.A.; Reddy, E.S. Deep intrusion net: An efficient framework for network intrusion detection using hybrid deep TCN and GRU with integral features. *Wirel. Netw.* **2025**, *31*, 1255–1278.
20. Ogunseyi, T.B.; Thiyagarajan, G. An Explainable LSTM-Based Intrusion Detection System Optimized by Firefly Algorithm for IoT Networks. *Sensors* **2025**, *25*, 2288. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Benmalek, M.; Seddiki, A. Particle swarm optimization-enhanced machine learning and deep learning techniques for Internet of Things intrusion detection. *Data Sci. Manag.* **2025**, *8*, 423–435. [\[CrossRef\]](#)
22. Grandhi, A.; Singh, S.K. Interrelated dynamic biased feature selection and classification model using enhanced gorilla troops optimizer for intrusion detection. *Alex. Eng. J.* **2025**, *114*, 312–330. [\[CrossRef\]](#)
23. AbuAladas, F.E.; Shehab, M.; Israwah, R.; Jaradat, G.; Qawqzeh, Y. A New Hybrid Approach for Improving Intrusion Detection System Based on Scatter Search Algorithm and Support Vector Machine. *J. Comput. Cogn. Eng.* **2025**, *5*, 143–151. [\[CrossRef\]](#)
24. Kumar, S. An enhanced whale optimizer based feature selection technique with effective ensemble classifier for network intrusion detection system. *Peer-to-Peer Netw. Appl.* **2025**, *18*, 4.
25. Anagha, A.; Thomas, C.; Balakrishnan, N. Optimized intrusion predictions through feature selection methods. *Comput. Secur.* **2025**, *157*, 104541. [\[CrossRef\]](#)

26. Umer, M.; Tahir, M.; Sardaraz, M.; Sharif, M.; Elmannai, H.; Algarni, A.D. Network intrusion detection model using wrapper based feature selection and multi head attention transformers. *Sci. Rep.* **2025**, *15*, 28718. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [\[CrossRef\]](#)
28. Moustafa, N. A New Distributed Architecture for Evaluating AI-Based Security Systems at the Edge: Network TON_IoT Datasets. *Sustain. Cities Soc.* **2021**, *72*, 102994. [\[CrossRef\]](#)
29. Booi, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; den Hartog, F.T.H. ToN IoT—The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Datasets. *IEEE Internet Things J.* **2021**, *9*, 485–496.
30. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *Ieee Access* **2020**, *8*, 165130–165150. [\[CrossRef\]](#)
31. Moustafa, N.; Keshky, M.; Debiez, E.; Janicke, H. Federated TON_IoT Windows datasets for evaluating AI-based security applications. In *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*; IEEE: Piscataway, NJ, USA, 2020; pp. 848–855.
32. Moustafa, N.; Ahmed, M.; Ahmed, S. Data Analytics-Enabled Intrusion Detection: Evaluations of ToN_IoT Linux Datasets. In *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*; IEEE: Piscataway, NJ, USA, 2020; pp. 727–735.
33. Moustafa, N. New Generations of Internet of Things Datasets for Cybersecurity Applications Based Machine Learning: TON_IoT Datasets. In *Proceedings of the eResearch Australasia Conference, Brisbane, Australia, 21–25 October 2019*.
34. Moustafa, N. A systemic IoT-Fog-Cloud architecture for big-data analytics and cyber security systems: A review of fog computing. *arXiv* **2019**, arXiv:1906.01055.
35. Ashraf, J.; Keshk, M.; Moustafa, N.; Abdel-Basset, M.; Khurshid, H.; Bakhshi, A.D.; Mostafa, R.R. IoTBoT-IDS: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities. *Sustain. Cities Soc.* **2021**, *72*, 103041. [\[CrossRef\]](#)
36. Carrier, T.; Victor, P.; Tekeoglu, A.; Lashkari, A.H. Detecting Obfuscated Malware Using Memory Feature Engineering. In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP)*, Online, 9–11 February 2022.
37. Lashkari, A.H.; Kaur, G.; Rahali, A. DIDarknet: A Contemporary Approach to Detect and Characterize the Darknet Traffic using Deep Image Learning. In *Proceedings of the 10th International Conference on Communication and Network Security, Tokyo, Japan, 27–29 November 2020*.
38. Apache Software Foundation. Apache Spark MLlib: Machine Learning Library. 2026. Available online: <https://spark.apache.org/mllib/> (accessed on 2 June 2026).
39. Kennedy, J.; Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*; IEEE: Piscataway, NJ, USA, 1995; Volume 4, pp. 1942–1948.
40. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
41. Mohamed, R.H.; Mosa, F.A.; Sadek, R.A. Efficient intrusion detection system for IoT environment. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 4. [\[CrossRef\]](#)
42. Yaras, S.; Dener, M. IoT-based intrusion detection system using new hybrid deep learning algorithm. *Electronics* **2024**, *13*, 1053. [\[CrossRef\]](#)
43. Dener, M.; Ok, G.; Orman, A. Malware detection using memory analysis data in big data environment. *Appl. Sci.* **2022**, *12*, 8604. [\[CrossRef\]](#)
44. Roy, K.S.; Ahmed, T.; Udas, P.B.; Karim, M.E.; Majumdar, S. MalHyStack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. *Intell. Syst. Appl.* **2023**, *20*, 200283. [\[CrossRef\]](#)
45. Karagöl, H.; Erdem, O.; Akbas, B.; Soylu, T. Darknet traffic classification with machine learning algorithms and SMOTE method. In *Proceedings of the 2022 7th International Conference on Computer Science and Engineering (UBMK)*; IEEE: Piscataway, NJ, USA, 2022; pp. 374–378.
46. Akour, I.; Alauthman, M.; Nahar, K.M.; Almomani, A.; Gupta, B.B. Analyzing darknet traffic through machine learning and neucube spiking neural networks. *Intell. Conver. Netw.* **2024**, *5*, 265–283. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.