Augmented whale feature selection for IoT attacks: Structure, analysis and applications

Article *in* Future Generation Computer Systems · May 2020

6 authors, including:

Majdi Mafarja
Birzeit University

Ali Asghar Heidari
National University of Singapore

Maria Habib
University of Jordan

Hossam Faris
University of Jordan

Augmented Whale Feature Selection for IoT Attacks: Structure, Analysis and Applications

Majdi Mafarja[a], Ali Asghar Heidari[b,c], Maria Habib[d], Hossam Faris[d], Thaer Thaher[e,f], Ibrahim Aljarah[d]

[a]*Department of Computer Science, Birzeit University, Birzeit, Palestine*
[b]*School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran*
[c]*Department of Computer Science, School of Computing, National University of Singapore, Singapore*
[d]*King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan*
[e]*Department of Computer Science, Al-Quds University, Abu Dies, Jerusalem, Palestine*
[f]*Department of Computer Science and Engineering, Palestine Polytechnic University,Hebron, Palestine*

Abstract

Smart connected appliances expand the boundaries of the conventional Internet into the new Internet of Things (IoT). IoT started to hold a significant role in our life, and in several fields as in transportation, industry, smart homes, and cities. However, one of the critical issues is how to protect IoT environments and prevent intrusions. Attacks detection systems aim to identify malicious patterns and threats that cannot be detected by traditional security countermeasures. In literature, feature selection or dimensionality reduction has been profoundly studied and applied to the design of intrusion detection systems. In this paper, we present a novel wrapper feature selection approach based on augmented Whale Optimization Algorithm (WOA), which adopted in the context of IoT attacks detection and handles the high dimensionality of the problem. In our approach, we introduce the use of both V-shaped and S-shaped transfer functions into WOA and compare the superior variant with other well-known evolutionary optimizers. The experiments are conducted using N-BaIoT dataset; wherein, five datasets were sampled from the original dataset. The dataset represents real IoT traffic, which is drawn from the UCI repository. The experimental results show that WOA based on V-shaped transfer function combined with elitist tournament binarization method is superior over S-shaped transfer function and outperforms other well-regarded evolutionary optimizers based on the obtained average accuracy, fitness, number of features, running time and convergence curves. Hence, we can conclude that the proposed approach can be deployed in IoT intrusion detection systems.

Keywords: Internet of things, Whale optimization algorithm, Feature selection, Attacks detection, Classification

Email addresses: `mmafarja@birzeit.edu`, `mmafarjeh@gmail.com` (Majdi Mafarja), `as_heidari@ut.ac.ir`, `aliasghar68@gmail.com`, `aliasgha@comp.nus.edu.sg`, `t0917038@u.nus.edu` (Ali Asghar Heidari), `hossam.faris@ju.edu.jo` (Hossam Faris), `thaer.thaher@gmail.com` (Thaer Thaher), `aljarah@ju.edu.jo` (Ibrahim Aljarah)

1. Introduction

Owing to the massive proliferation of IoT devices, IoT security is a stepping stone that has growing attention. Hung [1] -Gartner research vice president- predicted that the volume of IoT would reach almost 20 billion devices by 2020. IoT is a network of Internet-connected devices that collects information about the surrounding environment using embedded sensors and communicate together to exchange, process, and store the data [2]. Nowadays, IoT devices are deployed in a wide range of applications as transportation, healthcare, and military. This vast number of smart connected devices brings new security and privacy challenges. However, computational and technical limitations are critical bottlenecks for improving the security and privacy of IoT. IoT is vulnerable to network and software attacks as well as to privacy leakage [2]. This has been observed as a large scale Distributed Denial of Service attack (DDoS) in 2016, conducted by Mirai botnet and targeted Dyn (Domain name system provider), which disrupted a large number of services like CNN, PayPal and Netflix [1]. Nonetheless, by 2020, 25% of the recognized attacks in enterprises are attacks targeting IoT systems [3].

IoT devices are equipped with low computational resources that make them relatively more comfortable to be flooded, and vibrant platform for performing attacks [4]. IoT is plug-and-play devices that make them highly vulnerable to brute force attacks; since they have default passwords by their manufacturing settings [2]. Mainly, IoT is vulnerable to a wide range of attacks like sinkhole attacks, wormhole attacks, selective forwarding attacks, Sybil attacks, hello flooding attacks, and DoS attacks [5]. Generally, IoT systems consist of three main layers; the perception layer, the network layer, and the application layer, where each layer exposed to different types of attacks [4]. In this paper, we emphasize on network layer attacks, mainly on TCP, UDP, ACK, and SYN flooding attacks. Owing to the limited computational-power of IoT environments, applying traditional security countermeasures like encryption and authentication is not effective [5]. Consequently, more robust security techniques are crucial. Researchers have developed new technologies to adapt to the heterogeneity and scalability of the IoT system and elevate its security. Such emergent technologies are like blockchain [6], fog and cloud computing techniques [7]. However, these techniques still encounter high time latency and scalability issues [6]. One of the common techniques to maintain the security of such networks is the use of Intrusion Detection Systems (IDSs). IDSs analyze the network traffic in order to identify malicious activities or attacks. Upon successful identification, the IDSs send a warning to the decision-making system, in order to take an action [8]. Typically, IDSs were classified based on the detection technique into several groups; statistical-based, machine learning and data mining-based, rule-based, and other [8].

In literature, there is a tendency towards data mining and machine learning techniques into IoT. Data mining is the process of extracting novel, intriguing, and potentially valuable knowledge from large amounts of data that is stored in databases, data warehouses, or information repositories [9]. The aim of data mining is not just creating a classification or description models of data that can best fit with it, but also to generalize with the new data [9]. However, analyzing IoT data in order to detect anomalies, outliers, frauds or predict traffic is a very complex and challenging task. This is because IoT data is high-dimensional and can be represented by hundreds to thousands of dimensions. This large number of features could include irrelevant, noise, and redundant features which can negatively affect the learning process in model development [10]. Moreover, with high-dimensional data, machine learning models require an enormous amount of training data, which this is well-known as the curse of dimensionality [11]. As a solution, the Feature Selection (FS) process can tackle this problem by choosing a set of relevant features. FS is one of the significant pre-processing tasks in machine learning and data mining. Evaluating the selected subset of features is done using filters or wrapper methods. Wrapper-based FS methods use an induction (learning) algorithm during the selection process [12]. On the other hand, filter-based approaches evaluate each feature independently on the induction algorithm [12]. The size of the search space is highly dependent on the number of features, thus searching for the best subset of features is considered as an exhaustive task [10]. A variety of search methods have been proposed to address the problem of searching for the optimal subset like Tabu search [13], sequential search [14], Harmony search [15], and Greedy search [14]. However, traditional search methods suffer from the stagnation in local optima or high computational cost [10].

The rapid developments in science and technology increase the complexity of different real-world problems [16, 17, 18]. Taking as an example the emergence of hard optimization problems, like energy consumption in IoT devices, transport fuel consumption, water distribution networks optimization, and many more [19, 20]. Roughly speaking, hard optimization problems are difficult to solve within the reasonable, deterministic amount of time [21, 22]. However, metaheuristic algorithms attempt to find an optimal approximation of the solution for hard optimization problems [21, 23]. As the name implies, a heuristic is a process of exploring and experiencing things [21]. Metaheuristic algorithms are mostly nature-inspired algorithms that use stochastic components as stated by Thaher et al. [24], Chen et al. [25, 26], Xu et al. [27]. Metaheuristics are popular in the field of FS for their excellent performance [12] and global search abilities [10].

WOA is a metaheuristic algorithm that mimics the foraging behavior of the humpback whales found in nature [28]. This paper aims to promote the performance of anomalies detection or traffic classification into normal or attacks for IoT environments, by addressing the curse of dimensionality of IoT traffic by proposing a wrapper-based feature selection method, which is based on augmented WOA. Since WOA is originally developed to deal with continuous problems; hence, we implement the main version of WOA [28] with a new modification to deal with binary problems, in which both the V-shaped and S-shaped transfer functions have been integrated into WOA. The superior variant of transfer function among all is compared with well-regarded evolutionary optimizers; which they are grasshopper optimizer, grey wolf optimizer, gravitational search algorithm, particle swarm optimizer, ant lion optimizer, bat algorithm, and the salp swarm algorithm. All the experiments were conducted on real-IoT data, where the IoT datasets were drawn from UCI repository [29]. A new five datasets have been constructed from the original data, in which, the training data contains and trained on two types of attacks. Whereas, the testing data contains the prior two attacks, besides eight new unseen attacks. The results reveal that the V-shaped transfer function with elitist tournament method is the superior binary WOA, overall other transfer function variants and over other evolutionary optimizers.

The rest of the paper is organized as follows. Section 2 is a review of related works. Section 3 presents the WOA algorithm. Section 4 discusses the proposed methodology.

Section 5 discusses the used datasets and their characteristics. Lastly, Section 6 presents the experimental results and analysis.

2. Review of related works

There are several practices have been proposed to analyze network traffic data. This section presents a review of metaheuristic algorithms for malicious traffic detection in IoT networks as well as about the deployment of evolutionary algorithms for feature selection tasks.

2.1. IoT IDSs based on metaheuristic algorithms

Several Metaheuristic algorithms were used in cybersecurity like an artificial neural network, swarm optimization algorithms, and genetic algorithm. Hamamoto et al. [30] proposed a network anomaly detection technique based on genetic algorithm and fuzzy logic. Whereas, Bin Ahmad et al. [31] used the genetic algorithm for detecting insider threats. Hajimirzaei and Navimipour [32] proposed a new intrusion Detection System (IDS) that filters network traffic into normal and malicious using Multilayer Perceptron, which is trained by applying an artificial bee colony algorithm. Another relevant study in [33], in which Ali et al. [33] designed a supervised IDS in order to detect new attacks, by using particle swarm optimization (PSO) to build a fast learning network. Also, Selvakumar and Muneeswaran [34] proposed IDS using Bayesian networks and C4.5 to classify the network traffic, in which they deployed the firefly algorithm to make the feature selection. Nonetheless, Panigrahi and Patra [35] built an efficient IDS using a layered model. Five rule-based classifiers were used alongside three evolutionary search methods (Ant search, genetic search, and PSO), where at each layer different search method is deployed. All the methods mentioned above were dedicated to classical network types and not particularly for IoT networks. Moreover, all the available IDSs are not convenient for the evolved IoT networks since they were developed to traditional Internet networks or typical wireless sensor networks [36]. Developing an IDS is a challenging task since IoT devices are accessible globally, have limited resources, use low-power links, and connected via untrusted IPv6 and 6LoWPAN network protocols [36]. To the best of our knowledge, there are few studies in the literature on using evolutionary algorithms for intrusion detection in IoT networks. However, Sanchez-Pi et al. [37] used Voronoi diagram-based Evolutionary Algorithm (VorEAl) for IoT intrusion detection. VorEAl evolves Voronoi diagrams that are used to classify IoT data into anomalous or normal. In which it represents the input as Voronoi cells. Particularly, VorEAl is convenient with IoT since it has low computational complexity. Despite the promising achieved results, but they intended to build a dataset that represents more real IoT traffic. Hodo et al. [38] proposed an approach based on artificial neural networks for intrusion detection and identifying DDoS attacks. The results achieved good performance; however, they used simulated IoT data. Greensmith [39] discussed how the problem of IoT security could be solved using the responsive artificial immune system. Additionally, he investigated the current immune inspired algorithms and how they can be modified to fit with the IoT system's requirements. Another effort by He et al. [40], where they discussed major challenges with emergent technologies (like supply chain and big data) in IoT networks and suggested how the use of evolutionary algorithms and computational intelligence can enhance and improve the security of IoT.

2.2. Metaheuristic based feature selections for IoT

Recently, evolutionary algorithms have been utilized in feature selection tasks and showed successful performance results [12]. To the best of our awareness, there is a lack of deploying evolutionary algorithms for protecting IoT as well as in implementing evolutionary feature selection to promote attack detection. However, Li et al. [41] proposed a wrapper-based FS for IoT intrusion detection systems. They utilized the Bat algorithm with Swarm Division and Binary Differential Mutation in order to select the relevant features. However, they did not use real-IoT data.

In addition, few studies have been conducted on using evolutionary algorithms for FS and intrusion detection in traditional networks. For instance, Xue et al. [42] adopted a self-adaptive differential evolution algorithm for FS in Wireless Sensor Networks (WSNs); in order to detect intrusions. Moreover, Liu et al. [43] utilized an improved social spider optimization (ISSO) algorithm for feature extraction and selection in WSNs intrusion detection. Another relevant study, where Popoola and Adewumi [44] designed wrapper-based FS for network intrusion detection, by using discretized differential evolution algorithm. Moreover, Guendouzi and Boukra [45] presented a new FS technique for intrusion detection using Biogeography Based Optimization (BBO) algorithm. Gharaee and Hosseinvand [46] proposed an anomaly IDS, in which they used the genetic algorithm for feature selection. Even that, Till now, no one used the WOA for feature selection in IDSs, neither in traditional networks nor in IoT networks.

Generally, different evolutionary algorithms have been applied for the feature selection process. For example, Zawbaa et al. [47] used Ant Lion optimizer. Emary et al. [48] utilized the Grey Wolf Optimizer. Moreover, the use of Particle Swarm Optimization (PSO) algorithm as in [49]. Additionally, the adoption of Ant Colony Optimization in [50]. Ghamisi and Benediktsson [51] designed a hybrid method of Genetic Algorithm (GA) and PSO for feature selection. Nonetheless, the integration of Grasshopper Optimization by Mafarja et al. [52]. The usage of the multi-verse optimizer algorithm for FS presented by Faris et al. [53]. Whereas, Faris et al. [54] deployed the Salp Swarm Algorithm. Additionally, Mafarja et al. [55] utilized Dragonfly Optimization alongside time-varying transfer functions for feature selection. Also, several works by Taradeh et al. [56], Aljarah et al. [57], Mafarja et al. [58], Zhang et al. [59], Faris et al. [60] utilized other competitive methods for this area.

Mostly, all the aforementioned studies were dedicated to traditional networks more than IoT networks. However, most of the studies devoted to IoT, use simulated network data. Nonetheless, few of them deployed the evolutionary algorithms for attack detection. For the first time, we are presenting the use of a whale optimization algorithm; in order to enhance attack detection in real IoT scenarios.

3. Preliminaries

3.1. Whale Optimizer

Mathematical modeling and simulation are two cores of many analytical methods [61, 62, 63, 64]. Long-term styles and behaviors of Humpback whales have been inspired by Mirjalili and Lewis [28] to develop a reliable population-based optimizer as one of the well-established models. In the WOA method, we have a set of whales, searching for the food source (quarry), and they move based on some spiral trajectories [65]. They also have some intelligent tactics such as bubble net attacking, which helps them to confuse the prey and then swim around him. In this method, the exploration phase is intensified enough and almost half of the iterations are devoted to the diversification phase. Then, this optimizer can focus on the exploitation phase. The next parts explain the different phases of WOA.

3.1.1. Exploitation step

To represent and mimic the hunting behaviors of search agents, the rules in Eqs. (1) and (2) are performed in each iteration [28].

$$D = \mid C.\vec{X}^*(t) - \vec{X}(t) \mid \tag{1}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A}.D \tag{2}$$

where t is iteration, X^* represents the best whale (leader) attained so far, X is the position of an agent, $\mid\ \mid$ presents the absolute value and $<.>$ is used to show an element by element multiplication [66]. The parameters of A and C can be realized by the rules in Eqs. (3) and (4).

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a} \tag{3}$$

$$\vec{C} = 2.\vec{r} \tag{4}$$

where we need to decrease $\overrightarrow{a}$ from 2 to 0 and $\overrightarrow{r}$ is a random number in [0,1] [67]. Referring to Eq. (2), the whales will move toward the prey. The parameter a is obtained using Eq. (3):

$$a = 2(1 - \frac{t}{L}) \tag{5}$$

where t is iteration and L is the upper bound of iterations. The helix-formed moving step is updated via rule in Eq. (6):

$$\vec{X}(t+1) = D'.e^{bl}.cos(2\pi l) + \vec{X}^*(t) \tag{6}$$

$$D' = \mid C.\vec{X}^*(t) - \vec{X}(t) \mid \tag{7}$$

where b is a constant value and l shows a random value inside [-1,1] [68]. Therefore, we have the following rule:

$$\vec{X}(t+1) = \begin{cases} Shrinking\ encircling\ via\ Eq.(2) & p < 0.5 \\ Spiral\ shaped\ path\ via\ Eq.(6) & p \geq 0.5 \end{cases} \tag{8}$$

where p is another random value inside [0,1].

3.1.2. Exploration step

The diversification phase is executed using rules in Eqs. (9) and (10).

$$D = \mid C.\vec{X_{rand}}(t) - \vec{X}(t) \mid \tag{9}$$

$$\vec{X}(t+1) = \vec{X_{rand}}(t) - \vec{A}.D \tag{10}$$

where the $\vec{X_{rand}}$ represents a search agent which is determined, randomly.

The pseudo-code of the described method is shown in Algorithm 1.

Algorithm 1 Pseudo-code of WOA

Initiate the parameters (e.g., maximum iterations L and number of agents)
Generate initial agents, randomly $X_i (i = 1, 2, \ldots, n)$
Obtain the fitness values
Set X^* as the best agent
while $(t < L)$ **do**
  for each whale **do**
    Update a, A, C, l, and p
    if $(p < 0.5)$ **then**
      if $(\mid A \mid < 1)$ **then**
        Update the current whale by Eq. (2)
      else if $(\mid A \mid > 1)$ **then**
        Select a random whale
        Update the agents using Eq. (10)
    else if $(p > 0.5)$ **then**
      Update the whales based on Eq. (6)
  Check the bounding conditions
  Obtain the fitness values
  Update X^*, if a better agent exists
  $t = t + 1$
return X^*

3.2. k-Nearest Neighbor (k-NN) Classifier

The k-NN technique is a well-established classifier, which is classified as a non-parametric and instance-based method. This approach can classify the datasets based on unlabeled instances. For this aim, it checks the distance between a selected instance and the other k instances in its neighborhood [69]. To evaluate the distance, we can utilize different rules studied in the previous works. The Euclidean distance is often used, which is shown in Eq. (11):

$$D(s_1, s_2) = (\sum_{i=1}^{n} (x_{1,i} - x_{2,i})^2)^{\frac{1}{2}} \tag{11}$$

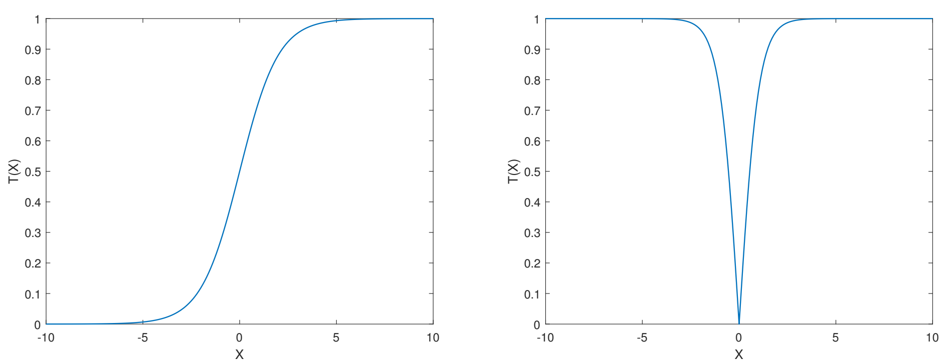
where s_1 and s_2 are points having n dimensions. In most of the wrapper FS methods, KNN is used to classify the datasets.

4. The proposed approach

Evolutionary algorithms were originally designed to tackle the continuous optimization problems [70, 71]. To deal with binary problems, they need to be converted efficiently. according to Crawford et al. [72], Transfer functions (TFs) are the most frequently used methods for this conversion. When using TFs for conversion purposes, two steps need to be noticed. In the first step, the TF is used to compute the probability of updating a corresponding dimension in the solution vector to 1 (selected) or 0 (not selected). While the second step, which called binarization method, is to update that dimension based on the resulted probability from the first step. In other words, the TF is applied on the real-values step vector produced by WOA such that a high probability of change is given to the dimension with a large value (which indicates that the current solution is far from the optimum solution obtained so far). While the dimension with small value, which means that the position of the processed solution is close to the best solution attained so far, is given a lower probability of being changed. A binarization rule is required to map the resulted intermediate solution into binary form.

In the literature, the TFs are categorized based on their shape into two main families: S-shaped and V-shaped functions. The sigmoid function, which belongs to S-shaped family, was firstly utilized by Kennedy and Eberhart [73] to propose a binary variant of PSO using Eq. (12), while Rashedi et al. [74] used tanh (V-shaped) to binarize the GSA algorithm using Eq. (13). These TFs are visualized in Fig.1.

According to Mirjalili and Lewis [75], different TFs have a major impact on the performance of the algorithm. They introduced six new variants of S-shaped and V-shaped TFs and investigated them with the PSO. The results revealed that the new introduced V-shaped TFs obtained the best results among all used TFs.



(a) (b)

Figure 1: (a) S-shaped and (b) V-shaped Transfer functions

$$T(x_j^i(t)) = \frac{1}{1 + e^{-x_j^i(t)}} \tag{12}$$

$$T(x_j^i(t)) = |\tanh(x_j^i(t))| \tag{13}$$

where x_i^j represents the j^{th} dimension of the i^{th} solution at iteration t, and $T(x_i^j(t))$ is the probability value obtained by TF.

For the binarization step, different methods have been incorporated with the above-mentioned TFs [72, 76]. The first binarization is the standard method used by Kennedy and Eberhart [73] with the S-shaped TF. According to Eq. (14), the dimension with a high probability value (outputted from the TF) is most likely to take the value of 1, otherwise, it is set to 0. Based on this concept, search agents that are far from the best solution are forced to take values of 1, while those that are very close to the best solution are forced to take values of 0. The disadvantage of this process is that it does not consider the current values of the solution and therefore, it leads to the premature convergence problem. The second most used binarization method (called complement method) was used by Rashedi et al. [74] with the V-shaped TF. In the complement method (as in Eq. (15)), the search agents are forced to stay in their current positions, when they are very close to the best solution, or flip to their complements when they move away from the best solution. This approach showed a high performance in handling various binary optimization problems [77, 75]. It overcomes the stagnation problem by exploring more search regions. However, this method does not consider the fittest solutions in the updating process.

$$X_i^k(t+1) = \begin{cases} 1 & r < T(x_j^i(t))) \\ 0 & Otherwise \end{cases} \tag{14}$$

$$X_i^k(t+1) = \begin{cases} \backsim X_i^j(t) & r < T(x_j^i(t))) \\ X_i^j(t) & Otherwise \end{cases} \tag{15}$$

where r is a random number in [0, 1] interval, $\backsim$ indicates the complement, and $X_i^j(t+1)$ is the new binary output.

Some improvements have been introduced based on the standard binarization rule. In the work of Crawford et al. [78, 79], the best solution so far was employed in the updating mechanism. Furthermore, Crawford et al. [80] proposed a binarization rule in which another solution selected via a roulette wheel selection method is employed. This method was presented as Eq. (16). It can be noticed that the agent which moves away from the best solution is re-positioned toward the fittest solution. While the agent that is close to the best solution is forced to take values of 0. However, The exploitation behavior in nature-inspired algorithms makes all search agents moving gradually towards the best solution. By considering this fact and based on the updating rule in Eq. (16), the probability for search agents to fall in local optimum becomes too high after some iterations.

$$X_i^k(t+1) = \begin{cases} X_*^k(t) & r < T(x_j^i(t))) \\ 0 & Otherwise \end{cases} \tag{16}$$

where X_*^k is the guide solution (the best as in Crawford et al. [78, 79], or selected by roulette wheel as in Crawford et al. [80])

Due to the drawbacks of the existing binarization approaches, we have proposed an augmented version of the complement method, in which stagnation problem of standard method and the exploration limitations of the complement method will be resolved. For this purpose, we utilized the evolutionary selection methods to select the guide solution which employed with the complement binarization rule to re-position the current solutions.

In this work, two-step binarization technique is used to convert the continuous search space to the WOA in binary form. In the first step, two basic TFs: S-shaped (*Sigmoid* TF) and V-shaped (*tanh* TF) are used to produce an intermediate vector where each element defines the probability of mutating the corresponding dimension in the position vector to 0 or 1. You can see rules in Eqs. (12) and (13), respectively

To transform the probability vector into a binary solution, six different binarization rules have been utilized in this paper to improve the convergence behavior of WOA: Standard (S) and Complement (C) are selected from the existing literature, while four new rules are introduced, namely Elitist (E), Elitist Tournament (ET), Elitist Roulette Wheel (ERW), and Elitist Rank (ER). Consequently, twelve versions of WOA (six per TF) are evaluated, which are WOA-S-S, WOA-S-C, WOA-S-E, WOA-S-ET, WOA-S-ERW, WOA-S-ER (for S-shaped TF), and WOA-V-S, WOA-V-C, WOA-V-E, WOA-V-ET, WOA-V-ERW, WOA-V-ER (for V-shaped TF). The general procedure of two-step binarization approach used in this research is illustrated in Fig. 2. The proposed approaches are explained in details in the following subsections.

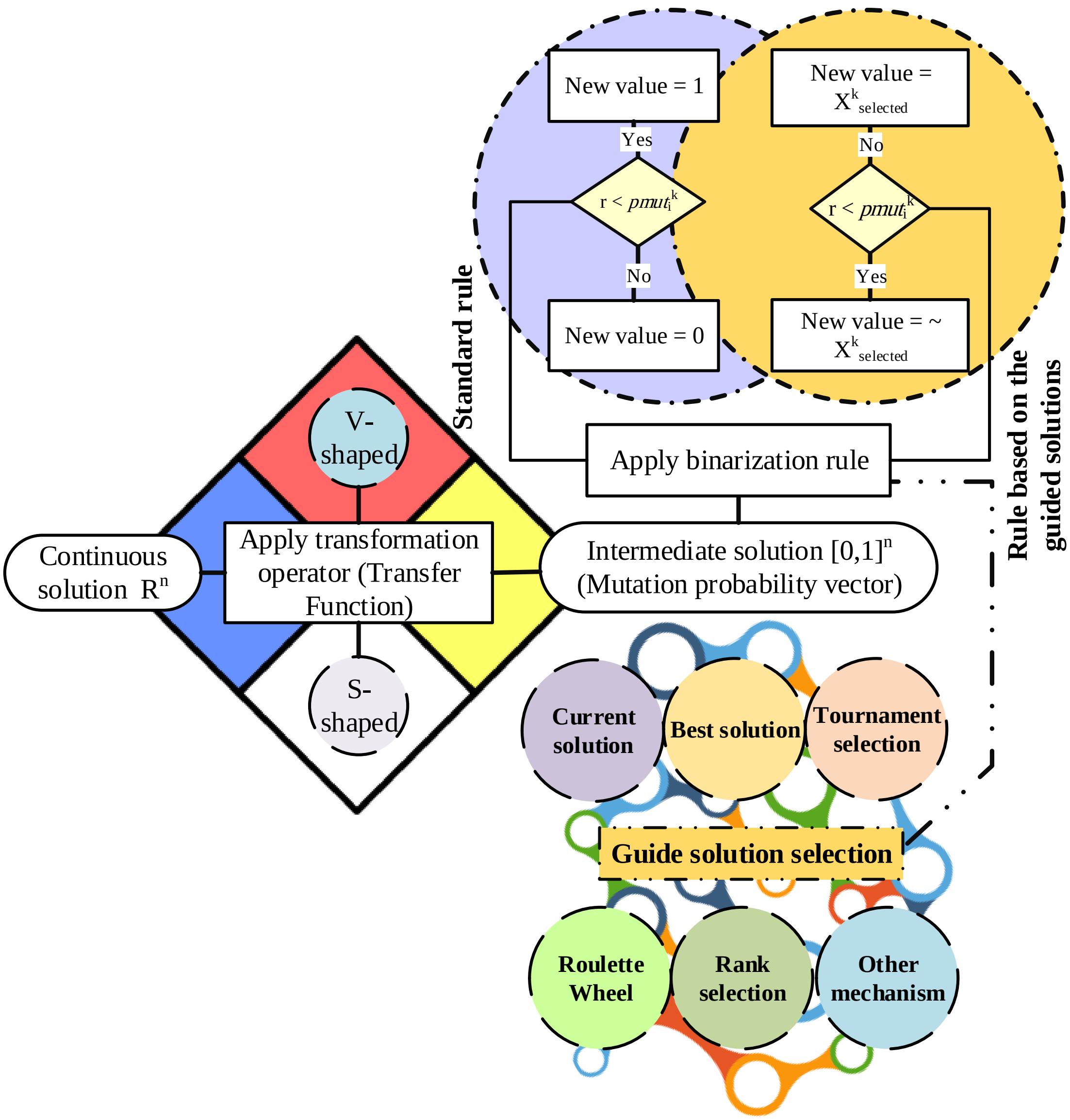


Figure 2: General procedure of two-step binarization method

4.1. Basic binarization methods for WOA

4.1.1. Standard method (WOA-S)

In WOA-S, the standard binarization rule is employed in WOA. This method was initially used by Kennedy and Eberhart [73] with the S-shaped TF. According to Eq. (14), the new value for each dimension of the binary solution is either set to 1 or 0 based on the corresponding probability value of intermediate solution regardless of the current value.

4.1.2. Complement method (WOA-C)

The complement rule explained in Eq. (15) is incorporated with WOA-C. This rule was originally introduced by Rashedi et al. [74]. The idea is that the values of the current processed position are randomly flipped or kept according to the calculated mutation probability.

4.2. Natural selection operators with binarization methods

As aforementioned in the necessary binarization procedures, it is clear that the other solutions are not considered for re-positioning the current ones. This limits the possibility of exploring more regions of the search space. Therefore, the traditional binarization mechanisms should be modified to benefit from the fittest properties of other solutions (not only the current solution) in the updating mechanisms and thus, overcoming the lack of population diversity.

Furthermore, the binarization behavior is identical to the binary mutation operator used in the evolutionary algorithms (EAs) where each value of the solution is switched between 1 and 0 or vice versa with a given mutation probability [76]. The selection process that determines which solutions are allowed to be a guide to re-position the current solutions is an important operator in EAs that ensures the balance between exploratory and exploitative potentials [81, 82]. This idea motivated our attempts to utilize various selection schemes to be employed with the discretization methods.

Different techniques are proposed to implement selection mechanism, the major three types are: tournament [83], roulette wheel (or proportional) [84], and rank-based selection [85]. Selection methods are based on the principle of "survival of the best" [86], where the fittest solution has a higher chance of being selected and thus leads to a better population (intensification). However, the worst solutions are not discarded and have a lower chance of being selected (diversification). The major factor that affects the efficiency of the selection process is called 'selective pressure', which can be defined as the tendency to select the fittest individuals of the current population [87]. The amount of selective pressure affects the balance between intensification and diversification. That is, too much pressure will introduce a bias toward the fittest solutions that will cause a lack of diversity and premature convergence, while a small amount of pressure maintains diversity and slows the convergence.

In this work, four binarization rules are proposed based on four different selection schemes for the guide solution used to update the current one. Those selection schemes are the fittest solution (Elitist), elitist tournament (ET), elitist roulette wheel (ERW), and elitist rank (ER). Equation (17) illustrates the general procedure for re-positioning the current processed solution according to the selected one ($X_{selected}$). The dimension with a high mutation probability (as calculated using the TF) has a higher chance to be the complement of the corresponding dimension of the selected solution, while the dimension with a low mutation probability has a higher chance to be set to the actual value of the corresponding dimension of the selected solution. The proposed variants of binarization methods for WOA will be discussed in the following subsections.

$$X^k_{new}(t+1) = \begin{cases} \backsim X^K_{selected}(t) & r < T(v^k_i(t+1)) \\ X^K_{selected}(t) & Otherwise \end{cases} \tag{17}$$

4.2.1. Elitist-based WOA (WOA-E)

The discretization method of WOA-E utilizes the fittest solution obtained so far to produce the new solution. The procedure can be viewed as the solution being processed is replaced by the best one; then, each element value is randomly flipped or kept based on the mutation probability as in Eq. (17). In the elitist method, the fittest solution is only considered (selection probability equal to 1), while the others are discarded (selection probability equal to 0). This method generates a population with characteristics derived from the best solution and thus leads to accelerate the convergence behavior [82]. However, high exploitation in this method may cause a premature convergence problem.

4.2.2. Elitist roulette-based WOA (WOA-ERW)

In WOA-ERW, the binarization step is incorporated with a roulette wheel (RW) selection scheme to identify the guide solution in Eq. (17). RW method was originally introduced to the Genetic Algorithm (GA) by Holland et al. [84]. The key notion of this method is that each solution has a non-zero opportunity to be selected. The probability of selection given for the i^{th} individual (p_i) is proportional to its absolute fitness value ($f(x_i)$) as in Eq. (18).

$$p_i = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)} \tag{18}$$

where N is the population size, and $f(x_i)$ is calculated by Eq. (19) for the minimization problem: [88].

$$f(x_i) = \frac{1}{1+f(x_i)} \tag{19}$$

The process of RW can be presented as a spinning roulette wheel, which divided into segments with different sizes, where each individual occupies a segment proportional to its fitness value. The fittest individuals (i.e large segments) have a higher chance of being selected than the poor ones (i.e small segments). The general steps of the described method are shown in Algorithm 2.

The main advantage of the roulette wheel selection mechanism compared to the elitist mechanism is that all individuals have a chance to be chosen and thus preserve the diversity of the population. However, the outstanding solutions have a high selection pressure, which will introduce a bias towards the best solutions, especially in the early stages of the search process, and therefore lead to the problem of stagnation in local optima. Moreover, as the population converges (i.e have individuals with similar fitness values), it is difficult to identify a better solution [89].

Algorithm 2 Pseudo-code of general proportional roulette wheel

```
Set c=0, i=1
// calculate cumulative probability vector.
while (i ≤ N) do
    calculate the selection probability of each solution (p_i)
    c = c + p_i
    i = i + 1
generate random number r between 0 and the summation of probabilities (c[N])
set i=1
while (i ≤ N) do
    if (r ≤ c(i)) then
        selected_index = i
        break
    i = i + 1
return (selected_index)
```

4.2.3. Elitist rank-based WOA (WOA-ER)

The guide solution for the binarization method in WOA-ER is identified by utilizing the rank-based selection mechanism. Rank selection method was proposed by [85] as a variant of the roulette wheel method to overcome the premature convergence problem. Each individual has a chance to be chosen based on its rank rather than its fitness. The process starts by sorting the individuals according to their fitness so that the rank N (where N is the population size) is assigned to the best individual while rank one is assigned to the worst one. Thus, the rank of each individual i in the sorted list is mapped to its selection probability (p_i) using the expression in Eq. (20):

$$p_i = \frac{rank_i}{n \times (n-1)} \tag{20}$$

Once selection probabilities have been computed, the selection process is performed by the roulette wheel mechanism as in Algorithm 2. It is noticeable that depending on the rank instead of fitness values prevents the domination of outstanding solutions since all individuals always have the same selection probabilities, and thus avoid the early stagnation problem. However, this method may lead to slow convergence. Moreover, it is computationally expensive because the population needs to be sorted on every cycle [82].

4.2.4. Elitist tournament-based WOA (WOA-ET)

In WOA-ET, tournament selection mechanism proposed by Goldberg et al., [83] is employed with the discretization method to chose the guide solution. Due to its simplicity and efficiency, it is the most popular selection method in EAs [90]. This method can be considered as a two-step selection mechanism. In the first step, k individuals are selected randomly from the current population, where k referred to as tournament size. In the second step, the fittest solution among the competitive individuals in the tournament is selected. The main benefit of this scheme is that it preserves diversity by giving each individual an equal probability of being selected for the competition step. However, this may lead to slow convergence [90, 82].

The tournament size (k) is an essential factor that used to adjust the selection pressure and thus, trad-off between the exploitative and exploratory potentials [87]. Larger values of k (higher selection pressure) will cause a bias toward the best solutions (i.e intensification) while lower values of k (lower selection pressure) shift the search toward a random behavior (i.e diversification). However, determining the proper value of the k parameter is challenging and depends on the nature of the problem being solved [88]. It is worth mentioning that WOA-E is a special case of WOA-ET when $k = N$. The pseudo-code of how WOA-ET select the guide solution is shown in Algorithm 3:

Algorithm 3 Pseudo-code of tournament selection

```
//Tournament selection for one solution
Identify the tournament size k
r = generate random index within [0, N]
set best = r
set i = 2
while (i ≤ k) do
    r = generate random index within [0, N]
    if (fit(X_r) < fit(X_best)) then
        best = r
    i = i + 1
return (best)
```

5. Dataset description, characteristics and preparation

The dataset used in this work is called N-BaIoT, and it was collected from real network traffic of nine IoT devices [91]. The dataset before processing consists of several files where each file belongs to a device that contains the traffic of normal and attack packets. There are ten classes of attacks that were generated using two families of botnet attack codes from the Github repository (Mirai, and BASHLITE). Botnet attack is a type of DDOS attack, where the attacker uses a large number of IoT devices to participate in the DOS to overwhelm a specific target. This type of attack is hard to detect since the device keeps function normally, and the user or the owner of the device will not notice if his device is a part of an attack, in some cases, the device may suffer from a delay of its functionality. The conducted attacks stand for five attacks from Mirai botnet, and another five attacks from Gafgyt botnet. The Mirai botnet attacks are automatic scanning for vulnerable devices, Acknowledgement (Ack) packets flooding, Synchronize (Syn) packets flooding, User Datagram Protocol (UDP) packets flooding, and UDP flooding with fewer options. Whereas the Gafgyt botnet attacks are sending spam data, UDP flooding, Transmission Control Protocol (TCP) packets flooding and sending spam data with specified Internet Protocol (IP) address and port. Table 1 shows the distribution of normal and attack traffic in the nine devices. The ratio of normal traffic to attacks is 1 to 13; therefore, the dataset can be considered highly imbalanced. It can be also noticed that device 3 and 7 contain only one class of attacks.

Table 1: Distribution of normal traffic and attacks in the 9 IoT devices, represented by the number of instances (packets) in each file

	Normal	Mirai_udpPlain	Mirai_udp	Mirai_SYN	Mirai_Scan	Mirai_ack	gafgyt_tcp	gafgyt_udp	gafgyt_scan	gafgyt_junk	gafgyt_combo	Total attacks
File 1	40,395	81,982	237,665	122,573	107,685	102,195	92,141	105,874	29,849	29,068	59,718	968,750
File 2	13,110	87,368	151,481	116,807	43,192	113,285	95,021	104,791	27,494	30,312	53,012	822,763
File 3	34,692	0	0	0	0	0	101,536	103,933	28,120	29,797	53,014	316,400
File 4	160,137	80,808	217,034	118,128	103,621	91,123	92,581	105,782	27,859	28,349	58,152	923,437
File 5	55,169	56,681	156,248	65,746	96,781	60,554	104,510	104,011	29,297	30,898	61,380	766,106
File 6	91,555	53,785	158,608	61,851	97,096	57,997	89,387	104,658	28,397	29,068	57,530	738,377
File 7	46,817	0	0	0	0	0	97,783	110,617	27,698	28,305	58,669	323,072
File 8	42,784	78,244	151,879	125,715	45,930	111,480	88,816	103,720	27,825	28,579	54,283	816,471
File 9	17,936	84,436	157,084	122,479	43,674	107,187	98,075	102,980	28,572	27,413	59,398	831,298
Total	502,595	523,304	1,229,999	733,299	537,979	643,821	859,850	946,366	255,111	261,789	515,156	6,506,674

In this work, five datasets are sampled and prepared from the original dataset. In our sampling approach, we constructed the training set for each dataset in a way to have only two different types of attacks, whereas the testing set contains all the types of attacks (i.e ten types of attacks). In other words, the developed model will be trained using just two types of attacks in addition to the normal traffic and will be tested on ten. This approach will form more challenging for the model as eight types of attacks will not be presented for the model before the testing time. It is worth mentioning that the training set has a balanced class distribution. In contrast, the testing set is imbalanced and reflects the original class distribution, in which the ratio of normal traffic to malicious traffic is 1:13. Table 2 summarize the characteristics of the five sampled datasets along with their class distribution ratios.

Table 2: A Description of the distribution of data for both training & testing parts, as well as the distribution of normal and attacks instances (packets) for each the training & testing sets

	Training dataset			Testing dataset			
	Normal traffic	Attack traffic	Types of attacks	Normal traffic	Attack traffic	Normal to attack ratio	Types of attacks
Dataset1	1664	1664	{COMBO,UDP}	128	1600	1:13	All (10 types)
Dataset2	1664	1664	{TCP,UDP}	128	1600	1:13	All (10 types)
Dataset3	1664	1664	{SCAN,SYN}	128	1600	1:13	All (10 types)
Dataset4	1664	1664	{UDP,ACK}	128	1600	1:13	All (10 types)
Dataset5	1664	1664	{TCP,UDPPLAIN}	128	1600	1:13	All (10 types)

6. Experimental results and simulations

All experiments in this study are coded in MATLAB 2018 licensed software under the same computing system. Table 3 shows the details of the system and user environments.

6.1. Parameter settings

One of the necessary conditions for the experiments is to perform a fair comparison. For this goal, we established the same condition for all evaluations, and all peers are wrapper methods that involve a learning model. For all algorithms, we used KNN with $K = 5$ to ensure the simplicity of the tests.

Table 3: The detailed settings of the system

Name	Setting
Hardware	
CPU	Intel Core(TM) i5 processor
Frequency	3.1GHz
RAM	4GB
Hard drive	500 GB
Software	
Operating system	Windows 7
Language	MATLAB R2018a

6.2. Results and discussions

In this section, we deeply investigate the performance of the proposed variants with different binarization schemes. To recognize the best variant, all versions are substantiated on various datasets, and then, the results are compared in terms of the primary metrics. After detecting the best variant, it is compared with the other well-established methods from the literature. Details of experiments are presented in the next subsections.

6.2.1. Different variants with S-shaped TF

In this subsection, we assess the effectiveness of each variant using S-shaped TF. Hence, each method has a different binarization strategy, but the same S-shaped TF.

The average accuracy rate, number of features, fitness, and running time results of the proposed BWOA using S-Shaped TF with various binarization techniques are exposed in Tables 4-7.

As per accuracy rates in Table 4, we see that the WOA_S_E version shows the best results for all cases. Based on F-test results, the best method is WOA_S_E, followed by WOA_S_ERW, WOA_S_C, WOA_S_ET, WOA_S_ER, and WOA_S_S, respectively. There is a very close competition between the top three methods, according to the obtained rates for all cases.

Table 4: The average accuracy results for S-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_S_S	WOA_S_C	WOA_S_E	WOA_S_ERW	WOA_S_ET	WOA_S_ER
Data 1	AVG	0.9733	0.9861	**0.9881**	0.9829	0.9837	0.9824
	STD	0.0240	0.0098	0.0080	0.0116	0.0080	0.0108
Data 2	AVG	0.9861	0.9878	**0.9893**	0.9891	0.9875	0.9867
	STD	0.0026	0.0043	0.0052	0.0053	0.0042	0.0029
Data 3	AVG	0.9151	0.9183	**0.9238**	0.9197	0.9190	0.9176
	STD	0.0059	0.0039	0.0202	0.0056	0.0026	0.0039
Data 4	AVG	0.9194	0.9204	**0.9228**	0.9204	0.9203	0.9203
	STD	0.0008	0.0007	0.0111	0.0007	0.0007	0.0007
Data 5	AVG	0.9147	0.9168	**0.9175**	0.9170	**0.9175**	0.9166
	STD	0.0041	0.0036	0.0038	0.0039	0.0032	0.0033
Ranking	F-Test	1	3.9	**5.9**	4.3	3.8	2.1

Based on the average number of features in Table 5, we see that the WOA_S_E version outperforms other variants with satisfactory STD values for all datasets. Referring to final ranks, the WOA_S_C, WOA_S_ERW, WOA_S_S, WOA_S_ER, and WOA_S_ET have obtained the next stages, respectively.

Table 5: The average number of features for S-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_S_S	WOA_S_C	WOA_S_E	WOA_S_ERW	WOA_S_ET	WOA_S_ER
Data 1	AVG	54.8000	50.6000	**45.8333**	51.4333	54.3333	51.7333
	STD	6.9798	6.6312	5.6022	5.0150	5.7615	5.9996
Data 2	AVG	44.2333	46.7000	**42.3000**	49.1000	47.7667	46.5000
	STD	4.3840	7.3819	6.9933	6.4560	6.5951	6.0215
Data 3	AVG	54.0667	53.9667	**45.5667**	52.4000	54.3333	54.4333
	STD	8.6699	5.6231	6.0211	6.0663	4.0115	5.9346
Data 4	AVG	47.0000	49.4667	**44.6000**	51.2333	49.6667	50.4333
	STD	6.3300	6.1405	5.6300	6.3555	6.0988	5.0901
Data 5	AVG	56.5000	51.9667	**44.7333**	51.4667	53.8333	51.6333
	STD	12.0766	8.0921	7.8298	6.7606	5.4715	6.0257
Ranking	F-Test	4	3.2	**1**	3.8	4.8	4.2

If we observe the average fitness results in Table 6, the WOA_S_E version provides the fittest results compared to other variants. As per F-test results, the best approach with satisfactory STD rates is the WOA_S_E, followed by WOA_S_C, WOA_S_ET, WOA_S_ERW, WOA_S_ER, and WOA_S_S, respectively. It is seen that the WOA_S_C and WOA_S_ET show very competitive results.

Table 6: The average fitness results for S-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_S_S	WOA_S_C	WOA_S_E	WOA_S_ERW	WOA_S_ET	WOA_S_ER
Data 1	AVG	0.0312	0.0182	**0.0158**	0.0215	0.0209	0.0220
	STD	0.0237	0.0097	0.0078	0.0113	0.0080	0.0108
Data 2	AVG	0.0176	0.0162	**0.0143**	0.0151	0.0166	0.0172
	STD	0.0024	0.0039	0.0050	0.0050	0.0038	0.0026
Data 3	AVG	0.0888	0.0856	**0.0795**	0.0841	0.0849	0.0863
	STD	0.0054	0.0038	0.0202	0.0057	0.0027	0.0038
Data 4	AVG	0.0839	0.0831	**0.0804**	0.0833	0.0832	0.0833
	STD	0.0007	0.0005	0.0111	0.0006	0.0006	0.0007
Data 5	AVG	0.0894	0.0869	**0.0856**	0.0867	0.0864	0.0871
	STD	0.0034	0.0035	0.0038	0.0038	0.0031	0.0031
Ranking	F-Test	6	3	**1**	3.1	3	4.9

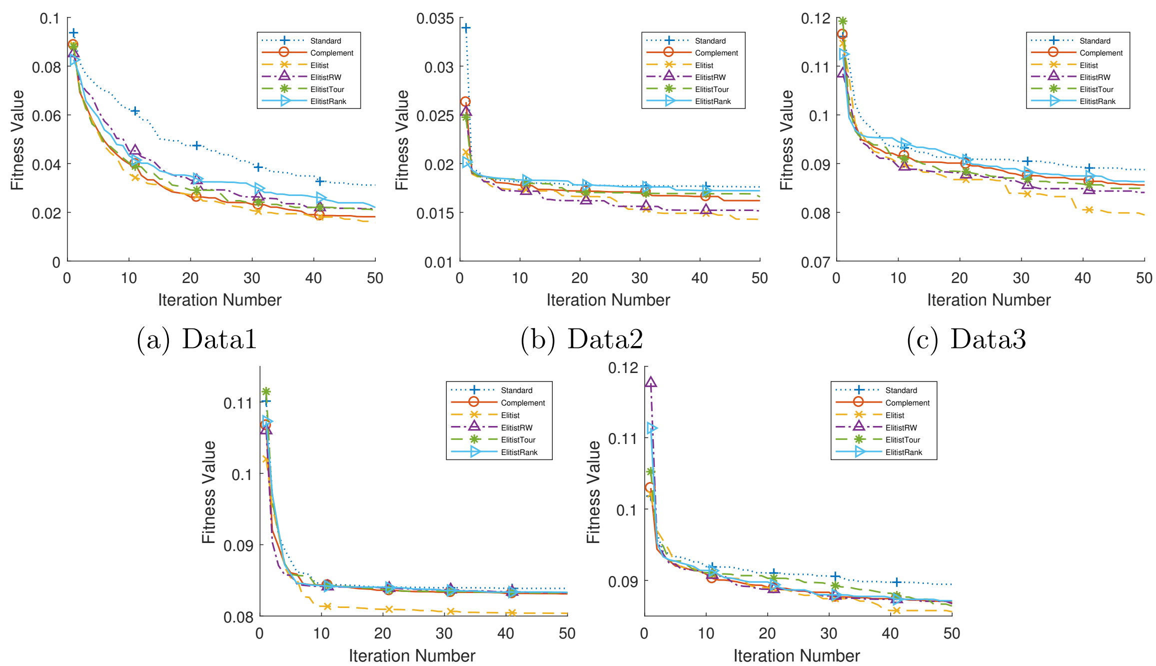
The time records in Table 7 disclose that the WOA_S_E is the fastest method, followed by WOA_S_ER, WOA_S_ERW, WOA_S_ET, WOA_S_C, and WOA_S_S methods, respectively.

Convergence behaviors of the proposed WOA-based methods with different binarization methods and S-shaped TF are demonstrated in Fig. 3. Based on the curves in Fig. 3, it is observed that the WOA_S_E is the best method in terms of convergence speed, while other peers almost reveal a competitive efficacy.

To investigate the significant differences between the results in terms of different metrics, we performed the Wilcoxon test. The obtained p-values are shown in Table 8. The reported

Table 7: The average running time for S-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_S_S	WOA_S_C	WOA_S_E	WOA_S_ERW	WOA_S_ET	WOA_S_ER
Data 1	AVG	1115.0790	507.8244	**388.1752**	505.4321	519.2185	506.8822
	STD	30.5503	23.5766	5.5141	22.4777	27.4476	22.5719
Data 2	AVG	1089.0823	515.7779	**408.7167**	501.1311	561.4072	493.7255
	STD	18.3113	24.1248	6.3923	25.1599	22.6492	6.4625
Data 3	AVG	979.4945	575.7358	**387.7059**	545.1944	498.2963	571.3142
	STD	46.9451	12.3219	9.5411	23.8602	10.2730	20.5155
Data 4	AVG	936.4449	546.9681	**387.1246**	501.9455	499.7265	509.2292
	STD	37.3279	22.4782	11.9758	21.7710	11.9152	27.0674
Data 5	AVG	1125.7411	529.0892	**395.9502**	581.6452	554.7243	494.9861
	STD	41.6083	21.7797	10.0535	19.0502	23.9955	10.8201
Ranking	F-Test	6	4.2	**1**	3.2	3.6	3



(a) Data1 (b) Data2 (c) Data3

(d) Data4 (e) Data5

Figure 3: Convergence curves of the proposed variants with different binarization methods and S-shaped TF

p-values in Table 8 show that the differences in results are significantly meaningful for most of the cases.

Table 8: p-values of the Wilcoxon test for the accuracy, number of features, fitness, and running time results of WOA-S-E and other methods for S-shaped TF (p≤0.05 are bolded)

	Accuracy					Features				
dataset	WOA_S	WOA_C	WOA_ERW	WOA_ET	WOA_ER	WOA_S	WOA_C	WOA_ERW	WOA_ET	WOA_ER
Data 1	**4.87E-03**	3.87E-01	5.90E-02	**4.34E-02**	**3.15E-02**	**3.57E-06**	**6.41E-03**	**1.78E-04**	**2.15E-06**	**4.00E-04**
Data 2	**2.04E-04**	1.48E-01	5.04E-01	**3.46E-02**	**1.40E-02**	1.49E-01	**1.58E-02**	**8.19E-04**	**4.91E-03**	**1.88E-02**
Data 3	**1.84E-02**	5.77E-01	9.47E-01	8.49E-01	1.87E-01	**2.40E-04**	**4.28E-06**	**1.84E-04**	**3.76E-07**	**3.85E-06**
Data 4	**3.34E-08**	**3.42E-02**	**3.42E-02**	**1.02E-02**	**1.02E-02**	3.46E-01	**4.62E-03**	**1.48E-04**	**3.12E-03**	**1.38E-04**
Data 5	**9.69E-03**	4.90E-01	7.72E-01	8.97E-01	3.58E-01	**4.83E-05**	**6.43E-04**	**3.71E-04**	**2.62E-06**	**8.72E-05**
	Fitness					Time				
Data 1	**5.55E-04**	2.20E-01	**1.80E-02**	**6.23E-03**	**1.03E-02**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 2	**3.11E-08**	**5.67E-04**	**2.09E-03**	**4.34E-06**	**2.84E-06**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 3	**1.28E-04**	**2.91E-02**	9.61E-02	**3.97E-02**	**3.02E-03**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 4	**1.87E-08**	**2.91E-05**	**1.65E-06**	**4.73E-06**	**3.71E-06**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 5	**4.06E-05**	5.64E-02	1.35E-01	1.05E-01	**4.05E-02**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**

6.2.2. Different variants with V-shaped TF

In this subsection, we assess the efficiency of each variant using V-shaped TF. Hence, each method has a different binarization strategy, but the same V-shaped TF.

The average accuracy rates, number of features, fitness results, and running time records of the BWOA using S-Shaped TF with various binarization techniques are shown in Tables 9-12.

As per the accuracy results of the developed WOA-based methods in Table 9, we see that the WOA_V_ET provides the fittest results, despite the WOA_V_S, which could not find a better place than the 6th-stage. The results of the F-test expose that the WOA_V_E, WOA_V_ERW, WOA_V_ER, WOA_V_C, and WOA_V_S are the next preferences in terms of accuracy metric, respectively. It is observed that the rates of all variants are very competitive.

According to the number of features, we observe that WOA_V_E is the best technique. The ranking results reveal that the WOA_V_C and WOA_V_S variants are the second and the third-best alternatives in terms of obtained relevant features.

As per fitness results, the best variant is WOA_V_ET, followed by WOA_V_E, WOA_V_ER, WOA_V_ERW, WOA_V_C, and WOA_V_S methods. These results show that the elitist tournament method can provide the best results, while the quality of results in the case of the standard method, which is known as the conventional binary WOA, is not very satisfying as the worst-ranked alternative. This observation emphasizes the significant role of the binarization technique on the excellence of binary WOA methods.

Based on the running times of variants, we see that the fastest performance of the proposed variants is experienced in the case of the elitist method, while the elitist roulette wheel method has led to the slowest speed. We also see that the binarization methods with a selection scheme are significantly slower than methods without a selection scheme, as it is expected.

The p-values in Table 13 also show that the detected differences in most of the cases and based on different metrics are significantly meaningful.

Table 9: The average Accuracy results for V-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_V_S	WOA_V_C	WOA_V_E	WOA_V_ERW	WOA_V_ET	WOA_V_ER
Data 1	AVG	0.9980	0.9986	0.9990	0.9999	**0.9999**	0.9998
	STD	0.0060	0.0053	0.0025	0.0003	0.0004	0.0004
Data 2	AVG	0.9927	0.9962	0.9981	**0.9987**	0.9978	0.9976
	STD	0.0065	0.0049	0.0033	0.0027	0.0043	0.0046
Data 3	AVG	0.9670	0.9895	0.9922	0.9850	0.9941	**0.9961**
	STD	0.0421	0.0208	0.0230	0.0312	0.0173	0.0144
Data 4	AVG	0.9732	0.9876	**0.9954**	0.9888	0.9952	0.9921
	STD	0.0350	0.0256	0.0115	0.0248	0.0172	0.0186
Data 5	AVG	0.9641	0.9800	0.9846	0.9806	**0.9890**	0.9854
	STD	0.0343	0.0234	0.0132	0.0267	0.0139	0.0212
Ranking	F-Test	1	2.2	4.4	3.9	**5.1**	4.4

Table 10: The average number of features for V-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_V_S	WOA_V_C	WOA_V_E	WOA_V_ERW	WOA_V_ET	WOA_V_ER
Data 1	AVG	2.5000	2.3000	**1.8333**	2.3333	2.2000	3.7333
	STD	0.9002	0.7497	0.4611	0.8841	0.7611	2.0331
Data 2	AVG	2.1667	**2.1333**	2.4333	3.4667	3.5333	4.5333
	STD	0.5921	0.6814	0.5683	2.0126	2.6876	2.3004
Data 3	AVG	2.7000	**2.1333**	2.5333	5.6000	2.6667	5.7333
	STD	1.3933	0.7761	1.1666	7.2474	1.5388	7.9217
Data 4	AVG	2.0667	2.1333	**2.0000**	2.6000	2.5333	4.0667
	STD	0.5833	0.5074	0.4549	1.6316	1.4320	2.2118
Data 5	AVG	2.4667	2.4000	**2.1000**	4.5667	4.4667	4.2000
	STD	1.2794	0.9685	0.8030	3.5398	3.4415	1.6484
Ranking	F-Test	3.2	2	**1.6**	4.8	3.8	5.6

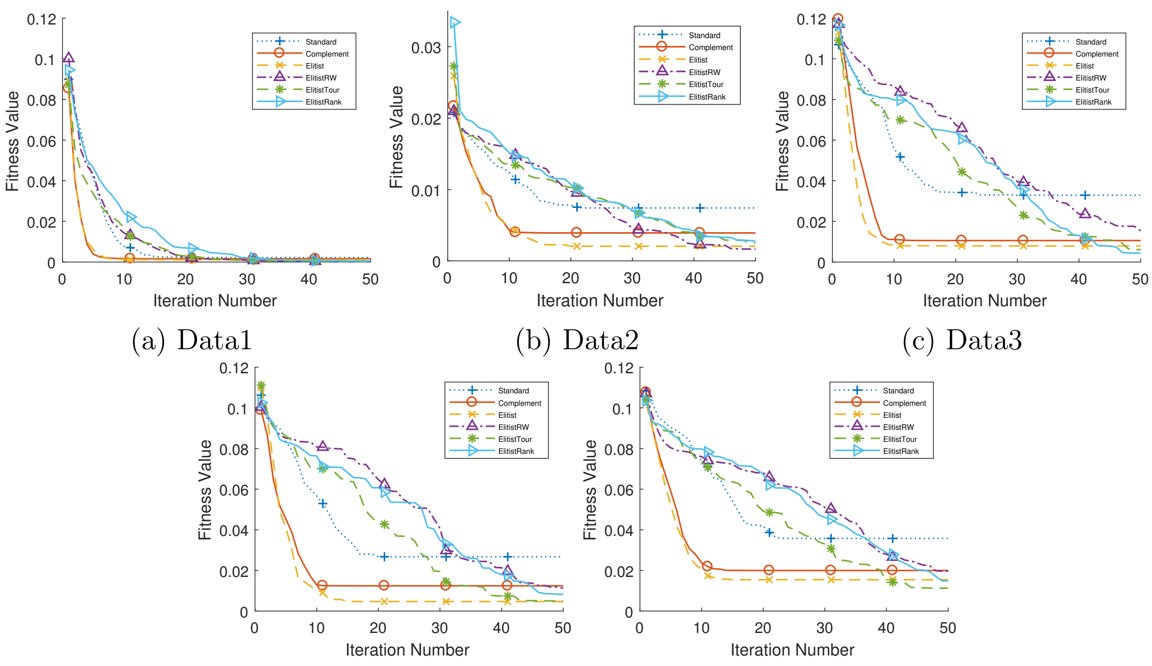
Table 11: The average fitness results for V-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_V_S	WOA_V_C	WOA_V_E	WOA_V_ERW	WOA_V_ET	WOA_V_ER
Data 1	AVG	0.0022	0.0016	0.0011	0.0003	**0.0003**	0.0005
	STD	0.0059	0.0052	0.0024	0.0004	0.0004	0.0005
Data 2	AVG	0.0074	0.0039	0.0020	**0.0016**	0.0025	0.0028
	STD	0.0064	0.0049	0.0033	0.0027	0.0042	0.0046
Data 3	AVG	0.0329	0.0106	0.0079	0.0153	0.0061	**0.0044**
	STD	0.0417	0.0206	0.0229	0.0314	0.0171	0.0149
Data 4	AVG	0.0267	0.0124	**0.0047**	0.0113	0.0049	0.0082
	STD	0.0347	0.0253	0.0114	0.0246	0.0171	0.0184
Data 5	AVG	0.0358	0.0200	0.0154	0.0196	**0.0113**	0.0148
	STD	0.0339	0.0232	0.0130	0.0265	0.0138	0.0210
Ranking	F-Test	6	4.8	2.6	3.1	**1.9**	2.6

Table 12: The average runing time for V-Shaped TF with various binarization techniques

Benchmark	Measure	WOA_V_S	WOA_V_C	WOA_V_E	WOA_V_ERW	WOA_V_ET	WOA_V_ER
Data 1	AVG	72.3882	36.2237	**34.9567**	125.8332	141.3283	178.4947
	STD	9.9035	2.9508	5.1691	21.6767	20.9900	28.3152
Data 2	AVG	70.8111	**38.6849**	41.5443	157.5332	158.8498	193.6005
	STD	6.7289	4.5432	4.9965	35.0779	28.7516	23.3072
Data 3	AVG	80.8584	44.0454	**39.5868**	201.5595	179.3935	208.5372
	STD	7.6494	5.3780	4.0352	39.7943	25.3290	30.8352
Data 4	AVG	85.2780	46.3031	**43.3901**	189.8144	169.7138	200.4964
	STD	10.1481	4.2200	5.8868	30.6690	25.5382	25.6574
Data 5	AVG	82.6338	46.6991	**44.0449**	195.6686	174.8223	201.8014
	STD	11.4962	6.8935	4.3402	44.0607	24.2564	23.9331
Ranking	F-Test	3	1.8	**1.2**	4.6	4.4	6

Convergence behaviors of all methods with V-shaped TF are demonstrated in Fig. 4. As curves show, the elitist and complement binarization methods present a better trend compared to other variants for all datasets, then, they have inertia to local optima; therefore, we see that the final results of binary WOA with elitist tournament method is slightly better than other peers.



(a) Data1 (b) Data2 (c) Data3

(d) Data4 (e) Data5

Figure 4: Convergence curves for WOA with different binarization methods for V-shaped TF

6.2.3. *Impact of TFs on WOA with each binarization technique*

To study the impact of TF on the excellence of results, Tables 14, (A.25, A.26, and A.27 at Appendix A) compare the variants with S-shaped and V-shaped TF and utilization of each binarization technique in terms of different metrics.

Table 13: p-values of the Wilcoxon test for the accuracy, number of features, fitness, and running time results of WOA-V-ET and other methods for V-shaped TF (p≤0.05 are bolded

	Accuracy					Features				
dataset	WOA_S	WOA_C	WOA_E	WOA_ERW	WOA_ER	WOA_S	WOA_C	WOA_E	WOA_ERW	WOA_ER
Data 1	**1.29E-02**	**1.41E-02**	**2.21E-02**	4.68E-01	4.53E-01	5.92E-02	3.29E-01	**1.14E-02**	4.80E-01	**7.45E-05**
Data 2	**3.70E-04**	1.41E-01	8.16E-01	4.26E-01	4.53E-01	**2.47E-03**	**2.69E-03**	5.05E-02	5.60E-01	**5.41E-03**
Data 3	6.74E-02	1.25E-01	5.71E-01	4.86E-01	6.23E-01	9.67E-01	1.88E-01	9.67E-01	6.55E-02	**8.79E-03**
Data 4	7.98E-02	3.46E-01	**3.47E-02**	9.60E-02	2.36E-01	9.75E-02	2.51E-01	**4.73E-02**	5.58E-01	**1.42E-03**
Data 5	**3.15E-03**	**4.33E-02**	5.52E-02	5.24E-01	8.71E-01	**6.47E-04**	**1.14E-03**	**1.93E-05**	7.57E-01	2.82E-01
	Fitness					Time				
Data 1	**4.02E-02**	**2.62E-02**	7.68E-02	4.92E-01	**4.89E-04**	**2.61E-10**	**3.02E-11**	**3.02E-11**	**4.86E-03**	**4.11E-07**
Data 2	**5.65E-04**	1.20E-01	7.48E-01	9.28E-01	3.85E-01	**3.02E-11**	**3.02E-11**	**3.02E-11**	8.07E-01	**1.09E-05**
Data 3	8.48E-02	1.90E-01	6.34E-01	5.39E-01	9.94E-01	**3.02E-11**	**3.02E-11**	**3.02E-11**	**6.97E-03**	**8.56E-04**
Data 4	3.44E-01	7.05E-01	1.37E-01	3.54E-01	**2.21E-02**	**3.34E-11**	**3.02E-11**	**3.02E-11**	**1.91E-02**	**3.59E-05**
Data 5	**3.39E-03**	5.35E-02	6.66E-02	5.34E-01	8.42E-01	**3.02E-11**	**3.02E-11**	**3.02E-11**	**1.27E-02**	**1.49E-04**

As per accuracy results, we see that the all variants with V-shaped TF are superior to alternative versions with S-shaped TF in dealing with 100% of datasets. For dataset 3-5, using V-shaped instead of S-shaped TF has led to more than 5-8% of improvements in the accuracy rates. The same observation is experienced in the case of the number of features, fitness values, and running time of methods. All of the variants with V-shaped TF have outperformed other peers with S-shaped TF in dealing with all datasets. We see there is a big gap between AVG and STD results of variants with V-shaped and S-shaped TF in terms of the number of features in all cases. As per fitness values, we detect that the V-shaped TF leads to superior results for all variants, in the case of any binarization method.

As per running time results, we observe that the variants with V-shaped TF are much faster than variants with S-shaped TF. For instance, in dealing with the dataset 5 with the standard binarization scheme, we need only 82.6338 (s) when using V-shaped TF, while in the case of S-shaped TF, the essential time increases to 1125.7411 (s). It can be seen that the most time-consuming scheme in the case of S-shaped TF is the standard method, whereas other binarization methods are faster, remarkably.

According to the results, first, we see that both TF and binarization techniques can change the quality of results, significantly, and the final set of features will be different based on the effectiveness of utilized binarization method. The main reason for the better performance of the methods with V-shaped functions is that they can perform a more smooth transition from exploration to exploitation. V-shaped TF can help the methods to aggressively explore the feature space and allocate higher mutation probabilities for both nearby and far optimal solutions. Typically, we observe that V-shaped variants are much faster than S-shaped variants, which is more desired for achieving real-time detection, especially when proposing an intrusion detection system. That can be interpreted by the V-shaped ability of smoothly moving from the exploration to exploitation; therefore, it can more efficiently locate the optimal solutions.

6.3. Comparison of top variants of WOA

In this part, we are interested in comparing the top variants in terms of different metrics. Tables 15 is dedicated to the comparison of only top variants WOA_S_E and WOA_V_ET in terms of the accuracy, the number of features, fitness, and running time. As per the results

Table 14: Comparison between S-shaped and V-shaped TF with each binarization technique based on the average accuracy

Benchmark	Measure	WOA_S		WOA_C		WOA_E		WOA_ERW		WOA_ET		WOA_ER	
		S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped
Data1	AVG	0.9733	**0.9980**	0.9861	**0.9986**	0.9881	**0.9990**	0.9829	**0.9999**	0.9837	**0.9999**	0.9824	**0.9998**
	STD	0.0240	0.0060	0.0098	0.0053	0.0080	0.0025	0.0116	0.0003	0.0080	0.0004	0.0108	0.0004
Data2	AVG	0.9861	**0.9927**	0.9878	**0.9962**	0.9893	**0.9981**	0.9891	**0.9987**	0.9875	**0.9978**	0.9867	**0.9976**
	STD	0.0026	0.0065	0.0043	0.0049	0.0052	0.0033	0.0053	0.0027	0.0042	0.0043	0.0029	0.0046
Data3	AVG	0.9151	**0.9670**	0.9183	**0.9895**	0.9238	**0.9922**	0.9197	**0.9850**	0.9190	**0.9941**	0.9176	**0.9961**
	STD	0.0059	0.0421	0.0039	0.0208	0.0202	0.0230	0.0056	0.0312	0.0026	0.0173	0.0039	0.0144
Data4	AVG	0.9194	**0.9732**	0.9204	**0.9876**	0.9228	**0.9954**	0.9204	**0.9888**	0.9203	**0.9952**	0.9203	**0.9921**
	STD	0.0008	0.0350	0.0007	0.0256	0.0111	0.0115	0.0007	0.0248	0.0007	0.0172	0.0007	0.0186
Data5	AVG	0.9147	**0.9641**	0.9168	**0.9800**	0.9175	**0.9846**	0.9170	**0.9806**	0.9175	**0.9890**	0.9166	**0.9854**
	STD	0.0041	0.0343	0.0036	0.0234	0.0038	0.0132	0.0039	0.0267	0.0032	0.0139	0.0033	0.0212
Ranking	W|T|L	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**

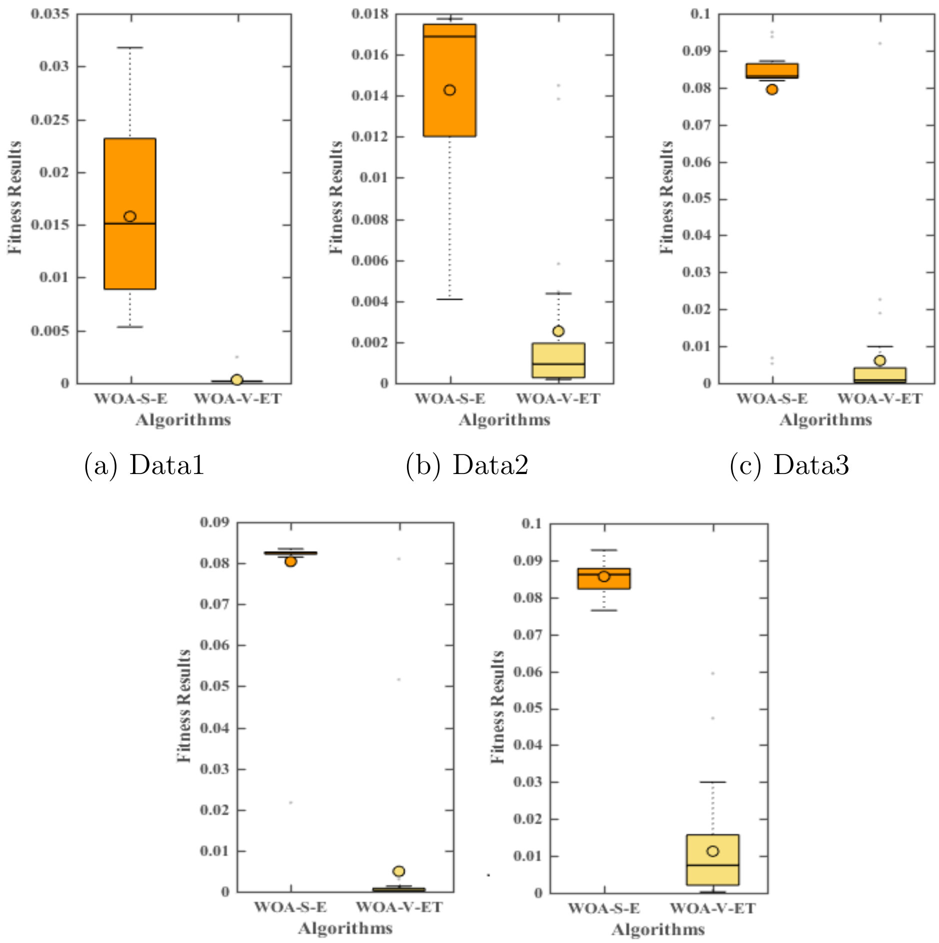
in Table 15, we see that WOA_V_ET is superior to the WOA_S_E in tackling all datasets. The WOA_V_ET approach can provide almost 99% of the classification rate with a faster running time. It provides almost 100% of accuracy for dataset 1.

Based on overall ranks, and results for each dataset, we conclude that using the elitist tournament method and V-shaped TF can generate the best binary WOA version. The superiority of the WOA_V_ET can be reasoned due to the use of V-shaped TF and higher exploration trends within the feature space. This mechanism can assist WOA_V_ET to escape convergence to any local optimum during iterations. In the case of any stagnation problem, it also has this capacity to jump out of them because of the enriched exploration trends. The other reason is because of the Elitist Tournament scheme, which helps the WOA_V_ET to avoid stagnation drawbacks in each step. The tournament scheme brings a higher chance to randomly select a better guiding solution instead of a global solution. In the case of any stagnation problem, the proposed method can help the algorithm to jump out of LO based on the tournament mechanism. Hence, in the rest of the experiments, we investigate the efficacy of WOA_V_ET as the best variant of the binary WOA.

Table 15: Comparison between the WOA-S-E and WOA-V-ET in terms of accuracy, number of features, fitness, and running time

Benchmark	Mesure	Accuracy		Number of Features		Fitness		Time	
		WOA-S-E	WOA-V-ET	WOA-S-E	WOA-V-ET	WOA-S-E	WOA-V-ET	WOA-S-E	WOA-V-ET
Data1	AVG	0.9881	**0.9999**	45.8333	**2.2000**	0.0158	**0.0003**	388.1752	**141.3283**
	STD	0.0080	0.0004	5.6022	0.7611	0.0078	0.0004	5.5141	20.9900
Data2	AVG	0.9893	**0.9978**	42.3000	**3.5333**	0.0143	**0.0025**	408.7167	**158.8498**
	STD	0.0052	0.0043	6.9933	2.6876	0.0050	0.0042	6.3923	28.7516
Data3	AVG	0.9238	**0.9941**	45.5667	**2.6667**	0.0795	**0.0061**	387.7059	**179.3935**
	STD	0.0202	0.0173	6.0211	1.5388	0.0202	0.0171	9.5411	25.3290
Data4	AVG	0.9228	**0.9952**	44.6000	**2.5333**	0.0804	**0.0049**	387.1246	**169.7138**
	STD	0.0111	0.0172	5.6300	1.4320	0.0111	0.0171	11.9758	25.5382
Data5	AVG	0.9175	**0.9890**	44.7333	**4.4667**	0.0856	**0.0113**	395.9502	**174.8223**
	STD	0.0038	0.0139	7.8298	3.4415	0.0038	0.0138	10.0535	24.2564
Ranking	W|T|L	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**

The Boxplots of fitness values for the best S-shaped and V-shaped variants are shown in Fig. 5 for all datasets. Also, the performance of WOA-V-ET versus WOA-S-E based on accuracy rates, the number of selected features, and fitness values are compared in Fig. 6.



(a) Data1 (b) Data2 (c) Data3

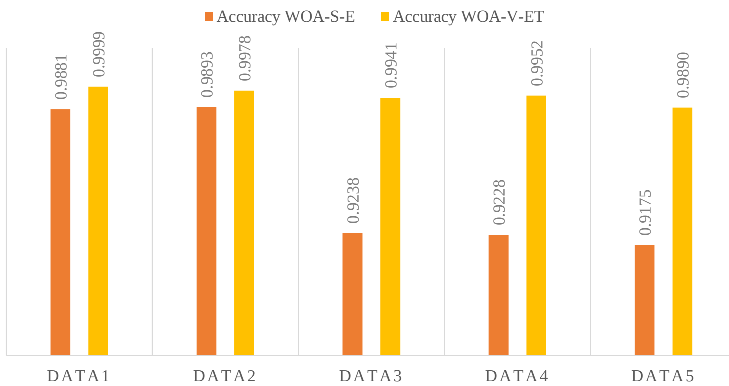
(d) Data4 (e) Data5

Figure 5: Boxplots for fitness values of the best S and V shaped variants for all datasets

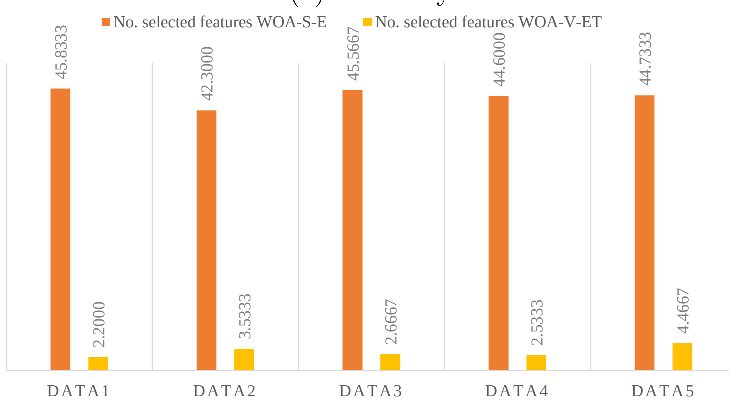
6.4. Comparison of WOA-V-ET with other optimizers

In this section, we compared the proposed WOA_V_ET variant with other well-established optimizers from literature in terms of different metrics. These experiments can reveal the core exploratory and exploitative merits of the proposed wrapper WOA_V_ET-based FS method compared to other existing wrappers in previous works. For this purpose, we compared the effectiveness of WOA_V_ET in terms of different measures with binary versions of Grasshopper Optimization Algorithm (GOA) [92], Grey Wolf Optimizer (GWO), Gravitational Search Algorithm (GSA), Particle Swarm Optimizer (PSO), Ant Lion Optimizer (ALO), Bat Algorithm (BAT/BA), and Salp Swarm Algorithm (SSA) [54]. These wrapper-based evolutionary FS approaches have recently revealed an excellent efficacy in dealing with different FS test problems and real-life cases. The initial parameters of these methods are set based on those recommended and set in the original papers.

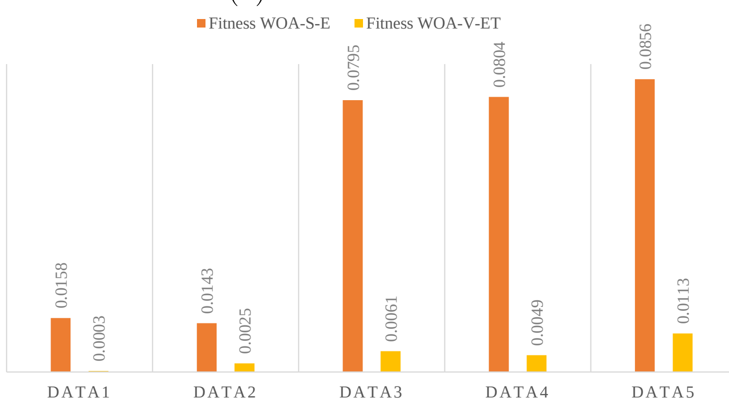
Table 16, and Tables (A.28, A.29, and A.30) at Appendix A, expose the results of the proposed WOA_V_ET versus all other approaches in terms of accuracy, number of features,



(a) Accuracy



(b) number of features



(c) Fitness

Figure 6: The performance of WOA-V-ET versus WOA-S-E in terms of accuracy rates, number of selected features, and fitness values

fitness, and running time values, respectively. Table 17 also reveals the significance of results based on the p-values of the statistical test. While Figure 7, indicates the convergence efficacy of the proposed WOA_V_E versus other peers. Tables (18-19) describe the most influential selected features among all datasets.

As per accuracy results in Table 16, it is detected that the proposed WOA_V_ET has attained the best AVG rates for all cases. If we check the F-Test ranks, it is clear that the best method is WOA_V_ET and the well-established BGOA, BPSO, bGWO, BSSA, bALO, BGSA, and BBA have attained the next ranks, respectively. For datasets 3-5, there is a substantial improvement in the accuracy rates compared to those realized by prior techniques.

Table 16: Comparison between WOA-V-ET and other optimizers based on average accuracy

Benchmark	Measure	WOA-V-ET	BGOA	bGWO	BGSA	BPSO	bALO	BBA	BSSA
Data1	AVG	**0.9999**	0.9989	0.9692	0.9193	0.9941	0.9263	0.8366	0.9465
	STD	0.0004	0.0006	0.0354	0.0222	0.0076	0.0152	0.1026	0.0263
Data2	AVG	**0.9978**	0.9901	0.9861	0.9802	0.9869	0.9855	0.8690	0.9856
	STD	0.0043	0.0058	0.0026	0.0247	0.0034	0.0000	0.1320	0.0003
Data3	AVG	**0.9941**	0.9264	0.9105	0.8914	0.9189	0.9079	0.7751	0.9117
	STD	0.0173	0.0252	0.0056	0.0251	0.0045	0.0024	0.1256	0.0053
Data4	AVG	**0.9952**	0.9378	0.9192	0.9051	0.9200	0.9190	0.7518	0.9193
	STD	0.0172	0.0327	0.0005	0.0227	0.0009	0.0000	0.1434	0.0007
Data5	AVG	**0.9890**	0.9253	0.9138	0.9016	0.9161	0.9115	0.7859	0.9134
	STD	0.0139	0.0217	0.0102	0.0298	0.0052	0.0025	0.1335	0.0035
Overall Ranking	F-Test	**8**	7	4.6	2	6	3	1	4.4

As per results for AVG number of features (Table A.29), we see that the WOA_V_ET is better than all methods. We see that the bGWO, BGOA, BPSO, BBA, BGSA, BSSA, and bALO have achieved the subsequent stages, respectively. It is vividly observed that the size of the feature set gotten by the WOA_V_ET is much smaller than those returned by other peers.

As per fitness rates, it is detected that the WOA_V_ET is the best method by fittest results compared to other competitors (Table A.30). F-Test results show that the BGOA is the second top algorithm, followed by the bGWO, BPSO, BSSA, bALO, BBA, and BGSA techniques.

The results indicate that the proposed WOA_V_ET has shown superior efficacy compared to other competitors. This can be reasoned owing to several main reasons: first, the higher intrinsic exploration potentials of the WOA compared to other optimizers. Besides, the proposed method utilizes the Elitist Tournament scheme, which alleviates the possible premature convergence and stagnation behaviors of the WOA_V_ET in each step, while optimizers such as BBA and BSSA cannot show a more stable performance. Also, the Elitist Tournament scheme helps WOA_V_ET to reach a more stable balance between exploration and exploitation trends.

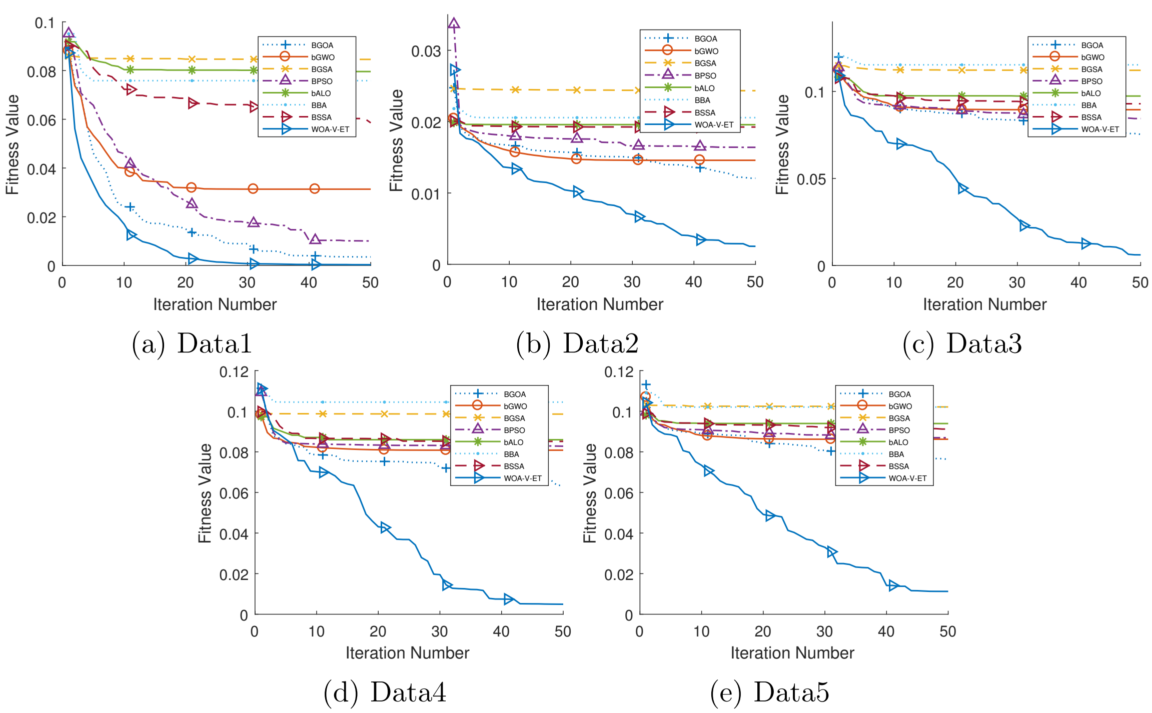
In terms of running time (Table A.28), the WOA_V_ET is the second-best optimizer, followed by BGOA, BBA, BPSO, BGSA, bALO, and BSSA techniques. It is seen that the bGWO shows a fast performance as well.

According to p-values in Table 17, it is vividly detected that the differences between the results of the proposed WOA_V_ET versus other peers are significantly meaningful in all cases.

Table 17: p-values of the Wilcoxon test for the accuracy, number of features, fitness, and running time results of WOA-V-ET and other optimizers ($p \leq 0.05$ are bolded)

	Accuracy							Features						
dataset	BGOA	bGWO	BGSA	BPSO	bALO	BBA	BSSA	BGOA	bGWO	BGSA	BPSO	bALO	BBA	BSSA
Data 1	**5.64E-09**	**2.28E-08**	**2.19E-12**	**1.69E-11**	**4.83E-13**	**3.48E-12**	**2.74E-12**	**3.12E-12**	**5.44E-12**	**3.08E-12**	**3.10E-12**	**3.06E-12**	**3.08E-12**	**3.07E-12**
Data 2	**5.03E-07**	**1.36E-10**	**5.61E-11**	**4.62E-09**	**1.21E-11**	**7.12E-11**	**2.20E-11**	**2.84E-11**	**9.05E-09**	**2.26E-11**	**2.24E-11**	**2.31E-11**	**2.30E-11**	**2.30E-11**
Data 3	**9.91E-10**	**4.11E-11**	**5.70E-11**	**7.61E-11**	**4.85E-12**	**3.23E-11**	**1.02E-10**	**1.56E-11**	**9.45E-11**	**1.53E-11**	**1.52E-11**	**1.56E-11**	**1.56E-11**	**1.53E-11**
Data 4	**8.12E-10**	**6.79E-12**	**2.50E-11**	**7.72E-11**	**2.02E-12**	**2.02E-11**	**1.71E-11**	**1.89E-11**	**9.37E-11**	**1.81E-11**	**1.82E-11**	**1.83E-11**	**1.80E-11**	**1.80E-11**
Data 5	**6.41E-10**	**7.91E-12**	**2.24E-11**	**2.21E-11**	**6.38E-12**	**2.92E-11**	**1.74E-11**	**3.54E-11**	**1.39E-07**	**2.58E-11**	**2.59E-11**	**2.61E-11**	**2.59E-11**	**2.62E-11**
	Fitness							Time						
Data 1	**9.92E-12**	**2.83E-11**	**5.20E-12**	**5.20E-12**	**5.15E-12**	**5.22E-12**	**5.20E-12**	**3.02E-11**	7.17E-01	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 2	**4.33E-09**	**8.90E-11**	**2.77E-11**	**7.38E-11**	**2.80E-11**	**2.82E-11**	**2.78E-11**	**3.02E-11**	1.41E-01	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 3	**7.57E-10**	**5.57E-11**	**4.58E-11**	**3.15E-10**	**2.76E-11**	**3.08E-11**	**8.16E-11**	**3.02E-11**	5.75E-02	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 4	**3.85E-10**	**1.53E-10**	**2.47E-11**	**2.45E-11**	**2.46E-11**	**2.47E-11**	**2.43E-11**	**3.02E-11**	**1.06E-03**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**
Data 5	**6.67E-10**	**3.57E-11**	**2.99E-11**	**2.95E-11**	**2.98E-11**	**2.99E-11**	**2.97E-11**	**3.02E-11**	7.01E-02	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**	**3.02E-11**

As per the curves in Fig. 7, we detect that the proposed WOA_V_ET can outperform previous methods in terms of convergence speed, and it shows a superior tendency to find a better solution faster than other methods. We see that the BGSA, bALO, and BBA algorithms cannot show a good enough performance to avoid local optima; hence, their potential to escaping local optima is not remarkable.



(a) Data1 (b) Data2 (c) Data3

(d) Data4 (e) Data5

Figure 7: Convergence curves for WOA-V-ET and other meta-heuristics

The WOA_V_ET can avoid immature convergence and show an accelerated convergence trend. The reason is that this method utilizes the Elitist tournament scheme, which increases the potential of the algorithm in enriching the excellence of found feature sets in a gradual manner. It also establishes a fair balance between the exploration in initial phases and exploitation in later steps. Hence, if we monitor the behavior of the proposed WOA_V_ET, a smooth convergence trend can be observed.

The selected features for Data1 dataset are shown in Table 18. The total number of selections for the most informative features are also shown in Table 19. The results in Table 19 are obtained after conducted 30 independent runs for each dataset (i.e the total number of runs is 150). By inspecting the results, it can be noticed that (H_L0.01_weight) feature achieved the best selection ratio (28.66%) followed by (MI_dir_L0.01_weight), which selected in 20% of total runs. The (H_L0.01_weight) feature is corresponding to the weight of the attack stream at a certain time and specific streaming source that is specified by an Internet Protocol (IP) address. Similarly, the (MI_dir_L0.01_weight) feature is the weight of the attacks packets stream at a specific time and specific streaming source that is defined by an IP and MAC addresses. For more information, Appendix A (Table A.24) shows a detailed description of the features of the dataset. This confirms that these two features are the most effective, therefore they have a high influence on the prediction accuracy of the learning model and should not be ignored for IoT Attacks data.

Table 18: The selected features for Data1 over 30 independent runs

Selected Features				Accuracy	No. features	FN	FP	Sensitivity	Specificity
H_L0.01_weight				0.99884	1	0	0	1	0.984375
MI_dir_L0.01_weight				0.99884	1	0	0	1	0.984375
H_L0.01_weight	HH_jit_L5_mean			1	2	0	0	1	1
MI_dir_L0.01_weight	HH_L5_mean			1	2	0	0	1	1
H_L0.01_weight	HH_L0.01_mean			1	2	0	0	1	1
MI_dir_L0.01_weight	HpHp_L1_magnitude			1	2	0	0	1	1
H_L0.01_weight	HpHp_L3_magnitude			1	2	0	0	1	1
H_L0.01_weight	HH_L3_magnitude			1	2	0	0	1	1
MI_dir_L0.01_weight	HpHp_L0.1_mean			1	2	0	0	1	1
H_L0.01_weight	HH_L0.1_mean			1	2	0	0	1	1
H_L0.01_weight	HH_L0.1_mean			1	2	0	0	1	1
H_L0.01_weight	HpHp_L3_mean			1	2	0	0	1	1
H_L0.01_weight	HH_L0.01_magnitude			1	2	0	2	1	1
H_L0.01_weight	HH_jit_L5_mean			1	2	0	0	1	1
H_L0.01_weight	HH_L1_mean			1	2	0	0	1	1
H_L0.01_weight	HH_L1_mean			1	2	0	0	1	1
H_L0.01_weight	HH_L0.01_magnitude			1	2	0	2	1	1
MI_dir_L0.01_weight	HpHp_L0.01_mean			1	2	0	0	1	1
H_L0.01_weight	HpHp_L0.01_magnitude			1	2	1	0	0.999375	1
MI_dir_L0.01_weight	HH_L0.01_magnitude			1	2	0	0	1	1
H_L0.01_weight	HH_L1_magnitude			1	2	0	0	1	1
H_L0.01_weight	HpHp_L5_mean			1	2	0	0	1	1
MI_dir_L0.01_weight	HH_L1_magnitude			1	2	0	0	1	1
H_L0.01_weight	HpHp_L5_magnitude			1	2	0	0	1	1
MI_dir_L0.01_weight	HpHp_L3_mean			1	2	0	0	1	1
MI_dir_L0.01_weight	HpHp_L5_radius	HpHp_L3_mean		1	3	0	0	1	1
MI_dir_L0.01_weight	HH_L1_radius	HpHp_L5_magnitude		1	3	0	2	1	1
MI_dir_L0.01_weight	HH_L1_weight	HpHp_L1_magnitude		1	3	0	0	1	1
MI_dir_L5_weight	MI_dir_L0.1_weight	HpHp_L3_std	HpHp_L0.01_mean	1	4	0	0	1	1
MI_dir_L3_weight	H_L3_weight	H_L3_mean	HpHp_L0.1_std	0.99884	4	0	0	1	0.984375
				0.99988	**2.20000**	**0.03333**	**0.20000**	**0.99998**	**0.99844**

Nonetheless, Table 20 exhibits the performance of WOA_V_ET against the Decision Trees algorithm (DT). In literature, the DT algorithm is used mainly for classification, where it creates a flowchart or tree-like structure of the classification rules. Each internal node denotes a feature, whereas the leaf nodes refer to the potential classification. To split the

Table 19: The total number of selections for the most informative features (selections ≥ 10)

Feature	Data1	Data2	Data3	Data4	Data5	Total
H_L0.01_weight	17	11	5	9	1	**43**
MI_dir_L0.01_weight	11	9	4	3	3	**30**
H_L1_weight	0	2	1	0	11	**14**
MI_dir_L1_weight	0	3	5	1	4	**13**
H_L0.1_weight	0	5	1	5	2	**13**
HH_L0.01_magnitude	3	8	0	2	0	**13**
MI_dir_L0.1_weight	1	3	5	2	0	**11**
MI_dir_L3_weight	1	2	3	2	2	**10**
HpHp_L3_mean	3	3	0	2	2	**10**

data into different classes, the decision tree algorithm selects the optimal feature based on some measures such as the Gini index or the information gain. Afterward, the algorithm stops depending on a predefined stopping criterion, for example, reaching a maximum size of the tree [93]. In fact, the DT algorithm has been adopted in various feature selection problems which demonstrated intriguing performance results [94]. However, according to Table 20, it is clear that WOA_V_ET algorithm outperformed the DT algorithm remarkably in terms of accuracy, False Negative (FN), and sensitivity. Nevertheless, it performed better at some datasets regarding the False Positive (FP) and the specificity. For instance, looking at the accuracy results, WOA_V_ET behaved considerably better than DT at most of the datasets, reaching a maximum of (0.999) at the first dataset. Regarding the FN, it showed a superb decreasing rate of misclassifying the normal as an anomaly, where the FN results are dramatically better than the DT algorithm. Similarly is for identifying the anomalies represented by the sensitivity measure. Evidently, WOA_V_ET accomplished the best in terms of sensitivity at Data1 reaching a value of 1.000. Furthermore, WOA_V_ET outperformed DT in terms of FP and specificity at Data2 and Data3, while for specificity, both WOA_V_ET and DT accomplished very close performance.

Table 20: A comparison between WOA_V_ET and DT algorithm based on accuracy, FP, FN, sensitivity, and specificity

Benchmark	Accuracy		FP		FN		sensitivity		specificity	
	WOA_V_ET	DT*	WOA_V_ET	DT*	WOA_V_ET	DT*	WOA_V_ET	DT*	WOA_V_ET	DT*
Data1	**0.9999**	0.9624	2.0000	**0.0000**	**0.0333**	65.0000	**1.0000**	0.9594	0.9846	**1.0000**
Data2	**0.9978**	0.7558	**1.4000**	2.0000	**2.4667**	420.0000	**0.9985**	0.7375	**0.9891**	0.9844
Data3	**0.9941**	0.7402	**2.7667**	82.0000	**7.4667**	367.0000	**0.9953**	0.7706	**0.9784**	0.3594
Data4	**0.9952**	0.7384	3.4667	**0.0000**	**4.7667**	452.0000	**0.9970**	0.7175	0.9729	**1.0000**
Data5	**0.9890**	0.8582	6.4667	**0.0000**	**12.5000**	245.0000	**0.9922**	0.8469	0.9495	**1.0000**

The observed trends vividly indicate that the WOA_V_ET reveals superior convergence drifts compared to other peers. Besides, stagnation behaviors of other methods can be observed when dealing with all datasets. According to these experiments, we conclude that the WOA_V_ET shows excellent efficacy compared to other well-regarded optimizers in literature. Since we are looking for an efficient, accurate, and usable intrusion detection system over IoT environments, WOA_V_ET proves spectacular abilities to distinguish normal behaviors from intrusions. Therefore, owing to WOA_V_ET's better detection performance and generalization ability, we highly recommend it as a deployable intrusion detection method for dealing with IoT scenarios.

6.5. Performance of WOA_ V_ET on general benchmarks

This section investigates the behavior of WOA_V_ET algorithm on several general-purpose datasets, as well as compares it with DT and KNN algorithms. The datasets were drawn from the UCI repository. Table 21 presents the used datasets, their names, number of features, and the number of instances. It is clear from Table 21 that the datasets are with a different number of features and instances.

Table 21: List of various UCI datasets

Dataset	No. of Features	No. of instances
Breastcancer	9	699
BreastEW	30	569
Exactly	13	1000
Exactly2	13	1000
HeartEW	13	270
Lymphography	18	148
M-of-n	13	1000
PenglungEW	325	73
SonarEW	60	208
SpectEW	22	267
CongressEW	16	435
IonosphereEW	34	351
KrvskpEW	36	3196
Tic-tac-toe	9	958
Vote	16	300
WaveformEW	40	5000
WineEW	13	178
Zoo	16	101

All conducted experiments in this section followed the same experimental settings of previous experiments; were all implemented on MATLAB 2018, withholding out 80% of the data for training and 20% for testing, while the k=5 for the KNN algorithm.

Table 22 reports the average accuracy of the proposed algorithm WOA_V_ET over the used UCI datasets, and against WOA_V_ET against WOA_V_S, WOA_V_C, KNN, and DT. WOA_V_ET achieved higher average classification accuracy on 50% of the datasets with the significant reasonable F-test values. Also, it can obtain most of the time an accuracy rate higher than 90% while reaching 100% accuracy at the Zoo dataset. However, the rest of the algorithms fails to achieve better than at most 17% of the datasets. On the other hand, Table 23 presents the average selected number of features for WOA_V_ET, WOA_V_S, and WOA_V_C regarding all datasets and against the original number of features of the datasets. Clearly, we can see that WOA_V_ET reduced the selected number of features for 56% of the datasets with significant F-test values. While WOA_V_S minimized the features' ratio to not more than 28% of the datasets, where WOA_V_C showed the weakest ability in reducing the features' ratio.

To sum up, WOA_V_ET algorithm achieved very-well on common general-purpose

Table 22: A comparison of average accuracy for WOA_V_ET against WOA_V_S, WOA_V_C, KNN, and DT over all datasets

Benchmark	with FS			without FS	
	WOA_V_S	WOA_V_C	WOA_V_ET	KNN	DT
Breastcancer	0.9750	0.9745	**0.9838**	0.9643	0.9500
BreastEW	0.9450	**0.9828**	0.9746	0.9386	0.9211
CongressEW	0.9797	0.9605	**0.9962**	0.9425	0.9425
Exactly	0.9043	0.7443	**0.9198**	0.6750	0.7100
Exactly2	0.7420	0.7550	0.7600	**0.7800**	0.6600
HeartEW	0.8426	0.8303	**0.8438**	0.8148	0.8148
IonosphereEW	0.9704	0.8953	**0.9779**	0.8451	0.9437
KrvskpEW	0.9609	0.9328	0.9559	0.9438	**0.9938**
Lymphography	**0.9526**	0.9092	0.8992	0.7333	0.7667
M-of-n	0.9605	0.8820	0.9740	0.8950	**1.0000**
penglungEW	0.9422	**0.9778**	0.9752	0.9231	0.3333
SonarEW	0.9437	0.9103	**0.9691**	0.9048	0.7857
SpectEW	**0.8883**	0.8728	0.8438	0.8148	0.8148
Tic-tac-toe	0.7971	0.8004	0.8325	**0.8646**	0.8490
Vote	0.9728	0.9794	**0.9806**	0.9533	0.9333
WaveformEW	0.7399	0.7194	0.7392	**0.7780**	0.7200
WineEW	0.9898	0.9815	**0.9954**	0.9722	0.8889
Zoo	**1.0000**	0.9984	**1.0000**	0.8571	**1.0000**
Rank (F-Test)	2.39	3.11	**1.83**	3.81	3.86

Table 23: A comparison of average selected number of features for WOA_V_ET against WOA_V_S, WOA_V_C over all datasets

Benchmark	WOA-V-S	WOA-V-C	WOA-V-ET	Actual No. of features
Breastcancer	3.83	**3.20**	3.53	9.00
BreastEW	7.53	8.07	**4.23**	30.00
CongressEW	3.70	2.37	**1.63**	16.00
Exactly	6.00	6.20	**5.20**	13.00
Exactly2	5.30	**1.00**	**1.00**	13.00
HeartEW	4.33	3.80	**2.30**	13.00
IonosphereEW	6.20	7.43	**4.17**	34.00
KrvskpEW	12.23	11.80	**8.03**	36.00
Lymphography	**4.33**	5.90	4.97	18.00
M-of-n	**5.83**	6.03	6.13	13.00
penglungEW	**8.47**	14.00	8.77	325.00
SonarEW	10.80	12.83	**8.93**	60.00
SpectEW	4.93	7.23	**1.30**	22.00
Tic-tac-toe	**4.63**	5.30	4.80	9.00
Vote	**2.03**	3.50	2.23	16.00
WaveformEW	12.87	11.77	**9.40**	40.00
WineEW	4.63	**3.20**	4.53	13.00
Zoo	4.63	**4.03**	4.23	16.00
Rank(F-test)	2.17	2.31	**1.53**	4

datasets, justifying it is reliable implementation and qualifying it for the objective of accurately identifying anomalies in a superior way over other algorithms.

7. Conclusion and future works

In this study, we proposed a augmented whale-inspired feature selection-based method for IoT attacks that is capable of being deployed in intrusion detection systems. In which, we tested both the S-shaped and V-shaped TFs with six different binarization methods for discretizing the continuous search space of WOA. We have created five new datasets by sampling the original N-BaIoT dataset. So, the algorithm is trained on two attacks and tested on ten attacks, where eight of them are new unseen attacks. In comparison with other well-regarded and recent algorithms, the binarization technique proved to be significant for the proposed optimizer efficiency. The Elitist tournament has ensured excellent capabilities in avoiding immature convergence, and in the potential of finding the optimal set of features in competitive time. Hence, we conclude that WOA_V_ET is able and worthy to be integrated within intrusion detection systems for IoT environments.

Future works can focus on the development of other binary optimizers such as harris hawks optimizer and investigate how efficient it can be. Evaluation of more classifiers and analyzing more datasets are also welcomed. In future works, we will extend the proposed framework for more variety of IoT datasets with different characteristics and scales.

Acknowledgment

This work is supported by project number 251166 granted by the Deanship of Scientific Research at Birzeit University. We also acknowledge the comprehensive comments of anonymous reviewers and efforts of editor in handling this research

References

[1] M. Hung, GARTNER Inc gartner insights on how to lead in a connected world, 2017.
[2] M. Conti, A. Dehghantanha, K. Franke, S. Watson, Internet of things security and forensics: Challenges and opportunities, 2018.
[3] S. Mishra, SAS Institute Inc iot security, 2017.
[4] R. Mahmoud, T. Yousuf, F. Aloul, I. Zualkernan, Internet of things (iot) security: Current status, challenges and prospective measures, in: Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for, IEEE, pp. 336–341.
[5] E. E.-D. Hemdan, D. Manjaiah, Cybercrimes investigation and intrusion detection in internet of things based on data science methods, in: Cognitive Computing for Big Data Systems Over IoT, Springer, 2018, pp. 39–62.
[6] M. A. Khan, K. Salah, Iot security: Review, blockchain solutions, and open challenges, Future Generation Computer Systems 82 (2018) 395–411.
[7] S. Prabavathy, K. Sundarakantham, S. M. Shalinie, Design of cognitive fog computing for intrusion detection in internet of things, Journal of Communications and Networks 20 (2018) 291–298.
[8] M. F. Elrawy, A. I. Awad, H. F. Hamed, Intrusion detection systems for iot-based smart environments: a survey, Journal of Cloud Computing 7 (2018) 21.
[9] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to data mining. 1st, 2005.
[10] B. Xue, M. Zhang, W. N. Browne, Particle swarm optimization for feature selection in classification: A multi-objective approach, IEEE transactions on cybernetics 43 (2013) 1656–1671.
[11] W. Qiao, W. Tian, Y. Tian, Q. Yang, Y. Wang, J. Zhang, The forecasting of pm2.5 using a hybrid model based on wavelet transform and an improved deep learning algorithm, IEEE Access 7 (2019) 142814–142825.
[12] M. Mafarja, S. Mirjalili, Whale optimization approaches for wrapper feature selection, Applied Soft Computing 62 (2018) 441–453.
[13] H. Zhang, G. Sun, Feature selection using tabu search method, Pattern recognition 35 (2002) 701–711.
[14] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (2014) 16–28.

[15] Z. W. Geem, J. H. Kim, G. V. Loganathan, A new heuristic optimization algorithm: harmony search, simulation 76 (2001) 60–68.
[16] J. Fan, D. Jiang, W. Liu, F. Wu, J. Chen, J. Daemen, Discontinuous fatigue of salt rock with low-stress intervals, International Journal of Rock Mechanics and Mining Sciences 115 (2019) 77–86.
[17] W. Qiao, B. Li, Z. Kang, Differential scanning calorimetry and electrochemical tests for the analysis of delamination of 3pe coatings, Int. J. Electrochem. Sci. 14 (2019) 7389–7400.
[18] W. Liu, Z. Zhang, J. Fan, D. Jiang, J. Daemen, Research on the stability and treatments of natural gas storage caverns with different shapes in bedded salt rocks, IEEE Access 8 (2020) 000507.
[19] S. Das, P. N. Suganthan, Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems, Jadavpur University, Nanyang Technological University, Kolkata (2010).
[20] W. Liu, Z. Zhang, J. Chen, J. Fan, D. Jiang, D. Jjk, Y. Li, Physical simulation of construction and control of two butted-well horizontal cavern energy storage using large molded rock salt specimens, Energy 185 (2019) 682–694.
[21] I. BoussaïD, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences 237 (2013) 82–117.
[22] G. Zhou, H. Moayedi, M. Bahiraei, Z. Lyu, Employing artificial bee colony and particle swarm techniques for optimizing a neural network in prediction of heating and cooling loads of residential buildings, Journal of Cleaner Production 254 (2020).
[23] G. Zhou, H. Moayedi, L. K. Foong, Teaching-learning-based metaheuristic scheme for modifying neural computing in appraising energy performance of building, Engineering with Computers 36 (2020).
[24] T. Thaher, A. A. Heidari, M. Mafarja, J. S. Dong, S. Mirjalili, Binary harris hawks optimizer for high-dimensional, low sample size feature selection, in: Evolutionary Machine Learning Techniques, Springer, 2020, pp. 251–272.
[25] H. Chen, A. A. Heidari, X. Zhao, L. Zhang, H. Chen, Advanced orthogonal learning-driven multi-swarm sine cosine optimization: Framework and case studies, Expert Systems with Applications 144 (2020) 113113.
[26] H. Chen, S. Li, A. A. Heidari, P. Wang, J. Li, Y. Yang, M. Wang, C. Huang, Efficient multi-population outpost fruit fly-driven optimizers: Framework and advances in support vector machines, Expert Systems with Applications 142 (2020) 112999.
[27] Y. Xu, H. Chen, A. A. Heidari, J. Luo, Q. Zhang, X. Zhao, C. Li, An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks, Expert Systems with Applications 129 (2019) 135 – 155.
[28] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in Engineering Software 95 (2016) 51–67.
[29] M. Lichman, UCI machine learning repository, 2013.
[30] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, M. L. Proença Jr, Network anomaly detection system using genetic algorithm and fuzzy logic, Expert Systems with Applications 92 (2018) 390–402.
[31] M. Bin Ahmad, A. Akram, M. Asif, S. Ur-Rehman, Using genetic algorithm to minimize false alarms in insider threats detection of information misuse in windows environment, Mathematical Problems in Engineering 2014 (2014).
[32] B. Hajimirzaei, N. J. Navimipour, Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm, ICT Express (2018).
[33] M. H. Ali, B. A. D. Al Mohammed, A. Ismail, M. F. Zolkipli, A new intrusion detection system based on fast learning network and particle swarm optimization, IEEE Access 6 (2018) 20255–20261.
[34] B. Selvakumar, K. Muneeswaran, Firefly algorithm based feature selection for network intrusion detection, Computers & Security 81 (2019) 148–155.
[35] A. Panigrahi, M. R. Patra, A layered approach to network intrusion detection using rule learning classifiers with nature-inspired feature selection, in: Progress in Computing, Analytics and Networking, Springer, 2018, pp. 215–223.
[36] S. Raza, L. Wallgren, T. Voigt, Svelte: Real-time intrusion detection in the internet of things, Ad hoc networks 11 (2013) 2661–2674.
[37] N. Sanchez-Pi, L. Martí, J. M. Molina, Applying voreal for iot intrusion detection, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, pp. 363–374.
[38] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, R. Atkinson, Threat analysis of iot networks using artificial neural network intrusion detection system, in: Networks, Computers and Communications (ISNCC), 2016 International Symposium on, IEEE, pp. 1–6.
[39] J. Greensmith, Securing the internet of things with responsive artificial immune systems, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, pp. 113–120.
[40] H. He, C. Maple, T. Watson, A. Tiwari, J. Mehnen, Y. Jin, B. Gabrys, The security challenges in the iot enabled cyber-physical systems and opportunities for evolutionary computing & other computational intelligence, in: Evolutionary Computation (CEC), 2016 IEEE Congress on, IEEE, pp. 1015–1021.
[41] J. Li, Z. Zhao, R. Li, H. Zhang, T. Zhang, Ai-based two-stage intrusion detection for software defined iot networks, IEEE Internet of Things Journal (2018).
[42] Y. Xue, W. Jia, X. Zhao, W. Pang, An evolutionary computation based feature selection method for intrusion detection, Security and Communication Networks 2018 (2018).
[43] L. Liu, B. Xu, X. Zhang, X. Wu, An intrusion detection method for internet of things based on suppressed fuzzy clustering, EURASIP Journal on Wireless Communications and Networking 2018 (2018) 113.
[44] E. Popoola, A. O. Adewumi, Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision., IJ Network Security 19 (2017) 660–669.
[45] W. Guendouzi, A. Boukra, Gab-bbo: Adaptive biogeography based feature selection approach for intrusion detection, International Journal of Computational Intelligence Systems 10 (2017) 914–935.
[46] H. Gharaee, H. Hosseinvand, A new feature selection ids based on genetic algorithm and svm, in: Telecommunications (IST), 2016 8th International Symposium on, IEEE, pp. 139–144.
[47] H. M. Zawbaa, E. Emary, B. Parv, Feature selection based on antlion optimization algorithm, in: Complex Systems (WCCS), 2015 Third World Conference on, IEEE, pp. 1–7.

[48] E. Emary, H. M. Zawbaa, A. E. Hassanien, Binary grey wolf optimization approaches for feature selection, Neurocomputing 172 (2016) 371–381.
[49] X. Wang, J. Yang, X. Teng, W. Xia, R. Jensen, Feature selection based on rough sets and particle swarm optimization, Pattern recognition letters 28 (2007) 459–471.
[50] Y. Chen, D. Miao, R. Wang, A rough set approach to feature selection based on ant colony optimization, Pattern Recognition Letters 31 (2010) 226–233.
[51] P. Ghamisi, J. A. Benediktsson, Feature selection based on hybridization of genetic algorithm and particle swarm optimization, IEEE Geoscience and Remote Sensing Letters 12 (2015) 309–313.
[52] M. Mafarja, I. Aljarah, A. A. Heidari, A. I. Hammouri, H. Faris, A.-Z. Ala'M, S. Mirjalili, Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems, Knowledge-Based Systems 145 (2018) 25–45.
[53] H. Faris, M. A. Hassonah, A.-Z. Ala'M, S. Mirjalili, I. Aljarah, A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture, Neural Computing and Applications 30 (2018) 2355–2369.
[54] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A.-Z. Ala'M, S. Mirjalili, H. Fujita, An efficient binary salp swarm algorithm with crossover scheme for feature selection problems, Knowledge-Based Systems 154 (2018) 43–67.
[55] M. Mafarja, I. Aljarah, A. A. Heidari, H. Faris, P. Fournier-Viger, X. Li, S. Mirjalili, Binary dragonfly optimization for feature selection using time-varying transfer functions, Knowledge-Based Systems 161 (2018) 185–204.
[56] M. Taradeh, M. Mafarja, A. A. Heidari, H. Faris, I. Aljarah, S. Mirjalili, H. Fujita, An evolutionary gravitational search-based feature selection, Information Sciences 497 (2019) 219–239.
[57] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang, S. Mirjalili, Asynchronous accelerating multi-leader salp chains for feature selection, Applied Soft Computing 71 (2018) 964–979.
[58] M. Mafarja, A. Qasem, A. A. Heidari, I. Aljarah, H. Faris, S. Mirjalili, Efficient hybrid nature-inspired binary optimizers for feature selection, Cognitive Computation (2019) 1–26.
[59] X. Zhang, Y. Xu, C. Yu, A. A. Heidari, S. Li, H. Chen, C. Li, Gaussian mutational chaotic fruit fly-built optimization and feature selection, Expert Systems with Applications 141 (2020) 112976.
[60] H. Faris, A. A. Heidari, A. M. Al-Zoubi, M. Mafarja, I. Aljarah, M. Eshtay, S. Mirjalili, Time-varying hierarchical chains of salps with random weight networks for feature selection, Expert Systems with Applications 140 (2020) 112898.
[61] W. Gao, J. L. G. Guirao, M. Abdel-Aty, W. Xi, An independent set degree condition for fractional critical deleted graphs., Discrete & Continuous Dynamical Systems-Series S 12 (2019) 877–886.
[62] W. Gao, H. Wu, M. K. Siddiqui, A. Q. Baig, Study of biological networks using graph theory, Saudi journal of biological sciences 25 (2018) 1212–1219.
[63] W. Gao, W. Wang, D. Dimitrov, Y. Wang, Nano properties analysis via fourth multiplicative abc indicator calculating, Arabian journal of chemistry 11 (2018) 793–801.
[64] W. Gao, J. L. Guirao, B. Basavanagoud, J. Wu, Partial multi-dividing ontology learning algorithm, Information Sciences 467 (2018) 35–58.
[65] W. Qiao, K. Huang, M. Azimi, S. Han, A novel hybrid prediction model for hourly gas consumption in supply side based on improved whale optimization algorithm and relevance vector machine, IEEE Access 7 (2019) 88218–88230.
[66] A. A. Heidari, I. Aljarah, H. Faris, H. Chen, J. Luo, S. Mirjalili, An enhanced associative learning-based exploratory whale optimizer for global optimization, Neural Computing and Applications (2019).
[67] J. Luo, H. Chen, A. A. Heidari, Y. Xu, Q. Zhang, C. Li, Multi-strategy boosted mutative whale-inspired optimization approaches, Applied Mathematical Modelling (2019).
[68] H. Chen, C. Yang, A. A. Heidari, X. Zhao, An efficient double adaptive random spare reinforced whale optimization algorithm, Expert Systems with Applications (2019) 113018.
[69] F. Pernkopf, Bayesian network classifiers versus selective k-nn classifier, Pattern Recognition 38 (2005) 1–10.
[70] W. Qiao, Z. Yang, Modified dolphin swarm algorithm based on chaotic maps for solving high-dimensional function optimization problems, IEEE Access 7 (2019) 110472–110486.
[71] W. Qiao, Z. Yang, Solving large-scale function optimization problem by using a new metaheuristic algorithm based on quantum dolphin swarm algorithm, IEEE Access 7 (2019) 138972–138989.
[72] B. Crawford, R. Soto, G. Astorga, J. García, C. Castro, F. Paredes, Putting continuous metaheuristics to work in binary search spaces, Complexity 2017 (2017).
[73] J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, volume 5, IEEE, pp. 4104–4108.
[74] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, Bgsa: binary gravitational search algorithm, Natural Computing 9 (2010) 727–745.
[75] S. Mirjalili, A. Lewis, S-shaped versus v-shaped transfer functions for binary particle swarm optimization, Swarm and Evolutionary Computation 9 (2013) 1–14.
[76] J. M. Lanza-Gutierrez, B. Crawford, R. Soto, N. Berrios, J. A. Gomez-Pulido, F. Paredes, Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization, Expert Systems with Applications 70 (2017) 67–82.
[77] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, Gsa: a gravitational search algorithm, Information sciences 179 (2009) 2232–2248.
[78] B. Crawford, R. Soto, R. Cuesta, M. Olivares-Suárez, F. Johnson Parejas, E. Olguin, Two swarm intelligence algorithms for the set covering problem, ICSOFT-EA 2014 - Proceedings of the 9th International Conference on Software Engineering and Applications (2014) 60–69.
[79] B. Crawford, R. Soto, M. Olivares-Suarez, W. Palma, F. Paredes, E. Olguin, E. Norero, A binary coded firefly algorithm that solves the set covering problem, Romanian Journal of Information Science and Technology 17 (2014) 252–264.

[80] B. Crawford, R. Soto, C. Peña, M. Riquelme-Leiva, C. Torres Rojas, F. Johnson Parejas, F. Paredes, Binarization methods for shuffled frog leaping algorithms that solve set covering problems, Advances in Intelligent Systems and Computing 349 (2015) 317–326.
[81] T. Blickle, L. Thiele, A comparison of selection schemes used in genetic algorithms, In Evolutionary Computation 11 (1996).
[82] N. Mohd Razali, J. Geraghty, Genetic algorithm performance with different selection strategies in solving tsp, volume 2.
[83] D. E. Goldberg, B. Korb, K. Deb, Messy genetic algorithms: Motivation, analysis, and first results, Complex Systems 3 (1990) 493–530.
[84] J. H Holland, Adaptation in natural and artificial systems, University of Michigan Press (1975).
[85] J. E. Baker, Adaptive selection methods for genetic algorithms, in: Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1985, pp. 101–111.
[86] E.-G. Talbi, Metaheuristics: from design to implementation, volume 74, John Wiley & Sons, 2009.
[87] T. Bäck, Selective pressure in evolutionary algorithms: A characterization of selection mechanisms (1996).
[88] M. Al-Betar, M. Awadallah, H. Faris, X.-S. Yang, A. T. Khader, O. Alomari, Bat-inspired algorithms with natural selection mechanisms for global optimization, Neurocomputing 273 (2017).
[89] A. Eiben, J. Smith, Introduction To Evolutionary Computing, volume 45, 2003.
[90] D. E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, volume 51.
[91] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, Y. Elovici, N-baiot: Network-based detection of iot botnet attacks using deep autoencoders, IEEE Pervasive Computing 13 (2018) 12–22.
[92] M. Mafarja, I. Aljarah, A. A. Heidari, A. I. Hammouri, H. Faris, A.-Z. Ala'M, S. Mirjalili, Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems, Knowledge-Based Systems 145 (2018) 25 – 45.
[93] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1986) 81–106.
[94] H. Rao, X. Shi, A. K. Rodrigue, J. Feng, Y. Xia, M. Elhoseny, X. Yuan, L. Gu, Feature selection based on artificial bee colony and gradient boosting decision tree, Applied Soft Computing 74 (2019) 634–642.

Appendix A.

Table A.24: Description of collected IoT features

Stream aggregation	
H	Source Internet Protocol (IP); stats summarizing the recent traffic from this packet's host (IP)
MI	Source MAC-IP; stats summarizing the recent traffic from this packet's host (IP + MAC)
HH	Channel; stats summarizing the recent traffic going from this packet's host (IP) to the packet's destination host.
HH-jit	Channel jitter; stats summarizing the jitter of the traffic going from this packet's host (IP) to the packet's destination host.
HpHp	Socket; stats summarizing the recent traffic going from this packet's host + port (IP) to the packet's destination host + port.
Time-frame (The decay factor Lambda)	
How much recent history of the stream is captured in these statistics	
L5, L3, L1, L0.1 and L0.01	
The statistics extracted from the packet stream	
weight	The weight of the stream
mean	The mean of the stream
std	The standard deviation of the stream
radius	The root squared sum of the two streams' variances
magnitude	The root squared sum of the two streams' means
cov	An approximated covariance between two streams
pcc	An approximated correlation coefficient between two streams

Table A.25: Comparison between S-shaped and V-shaped TF with each binarization technique based on the average number of features

Benchmark	Measure	WOA_S		WOA_C		WOA_E		WOA_ERW		WOA_ET		WOA_ER	
		S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped
Data1	AVG	54.8000	**2.5000**	50.6000	**2.3000**	45.8333	**1.8333**	51.4333	**2.3333**	54.3333	**2.2000**	51.7333	**3.7333**
	STD	6.9798	0.9002	6.6312	0.7497	5.6022	0.4611	5.0150	0.8841	5.7615	0.7611	5.9996	2.0331
Data2	AVG	44.2333	**2.1667**	46.7000	**2.1333**	42.3000	**2.4333**	49.1000	**3.4667**	47.7667	**3.5333**	46.5000	**4.5333**
	STD	4.3840	0.5921	7.3819	0.6814	6.9933	0.5683	6.4560	2.0126	6.5951	2.6876	6.0215	2.3004
Data3	AVG	54.0667	**2.7000**	53.9667	**2.1333**	45.5667	**2.5333**	52.4000	**5.6000**	54.3333	**2.6667**	54.4333	**5.7333**
	STD	8.6699	1.3933	5.6231	0.7761	6.0211	1.1666	6.0663	7.2474	4.0115	1.5388	5.9346	7.9217
Data4	AVG	47.0000	**2.0667**	49.4667	**2.1333**	44.6000	**2.0000**	51.2333	**2.6000**	49.6667	**2.5333**	50.4333	**4.0667**
	STD	6.3300	0.5833	6.1405	0.5074	5.6300	0.4549	6.3555	1.6316	6.0988	1.4320	5.0901	2.2118
Data5	AVG	56.5000	**2.4667**	51.9667	**2.4000**	44.7333	**2.1000**	51.4667	**4.5667**	53.8333	**4.4667**	51.6333	**4.2000**
	STD	12.0766	1.2794	8.0921	0.9685	7.8298	0.8030	6.7606	3.5398	5.4715	3.4415	6.0257	1.6484
Ranking	W|T|L	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**

Table A.26: Comparison between S-shaped and V-shaped TF with each binarization technique based on the average fitness

Benchmark	Measure	WOA_S		WOA_C		WOA_E		WOA_ERW		WOA_ET		WOA_ER	
		S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped
Data1	AVG	0.0312	**0.0022**	0.0182	**0.0016**	0.0158	**0.0011**	0.0215	**0.0003**	0.0209	**0.0003**	0.0220	**0.0005**
	STD	0.0237	0.0059	0.0097	0.0052	0.0078	0.0024	0.0113	0.0004	0.0080	0.0004	0.0108	0.0005
Data2	AVG	0.0176	**0.0074**	0.0162	**0.0039**	0.0143	**0.0020**	0.0151	**0.0016**	0.0166	**0.0025**	0.0172	**0.0028**
	STD	0.0024	0.0064	0.0039	0.0049	0.0050	0.0033	0.0050	0.0027	0.0038	0.0042	0.0026	0.0046
Data3	AVG	0.0888	**0.0329**	0.0856	**0.0106**	0.0795	**0.0079**	0.0841	**0.0153**	0.0849	**0.0061**	0.0863	**0.0044**
	STD	0.0054	0.0417	0.0038	0.0206	0.0202	0.0229	0.0057	0.0314	0.0027	0.0171	0.0038	0.0149
Data4	AVG	0.0839	**0.0267**	0.0831	**0.0124**	0.0804	**0.0047**	0.0833	**0.0113**	0.0832	**0.0049**	0.0833	**0.0082**
	STD	0.0007	0.0347	0.0005	0.0253	0.0111	0.0114	0.0006	0.0246	0.0006	0.0171	0.0007	0.0184
Data5	AVG	0.0894	**0.0358**	0.0869	**0.0200**	0.0856	**0.0154**	0.0867	**0.0196**	0.0864	**0.0113**	0.0871	**0.0148**
	STD	0.0034	0.0339	0.0035	0.0232	0.0038	0.0130	0.0038	0.0265	0.0031	0.0138	0.0031	0.0210
Ranking	W|T|L	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**

Table A.27: Comparison between S-shaped and V-shaped TF with each binarization technique based on the average running time

Benchmark	Measure	WOA_S		WOA_C		WOA_E		WOA_ERW		WOA_ET		WOA_ER	
		S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped	S-Shaped	V-Shaped
Data1	AVG	1115.0790	**72.3882**	507.8244	**36.2237**	388.1752	**34.9567**	505.4321	**125.8332**	519.2185	**141.3283**	506.8822	**178.4947**
	STD	30.5503	9.9035	23.5766	2.9508	5.5141	5.1691	22.4777	21.6767	27.4476	20.9900	22.5719	28.3152
Data2	AVG	1089.0823	**70.8111**	515.7779	**38.6849**	408.7167	**41.5443**	501.1311	**157.5332**	561.4072	**158.8498**	493.7255	**193.6005**
	STD	18.3113	6.7289	24.1248	4.5432	6.3923	4.9965	25.1599	35.0779	22.6492	28.7516	6.4625	23.3072
Data3	AVG	979.4945	**80.8584**	575.7358	**44.0454**	387.7059	**39.5868**	545.1944	**201.5595**	498.2963	**179.3935**	571.3142	**208.5372**
	STD	46.9451	7.6494	12.3219	5.3780	9.5411	4.0352	23.8602	39.7943	10.2730	25.3290	20.5155	30.8352
Data4	AVG	936.4449	**85.2780**	546.9681	**46.3031**	387.1246	**43.3901**	501.9455	**189.8144**	499.7265	**169.7138**	509.2292	**200.4964**
	STD	37.3279	10.1481	22.4782	4.2200	11.9758	5.8868	21.7710	30.6690	11.9152	25.5382	27.0674	25.6574
Data5	AVG	1125.7411	**82.6338**	529.0892	**46.6991**	395.9502	**44.0449**	581.6452	**195.6686**	554.7243	**174.8223**	494.9861	**201.8014**
	STD	41.6083	11.4962	21.7797	6.8935	10.0535	4.3402	19.0502	44.0607	23.9955	24.2564	10.8201	23.9331
Ranking	W|T|L	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**	0|0|5	**5|0|0**

Table A.28: Comparison between WOA-V-ET and other optimizers based on average running time

Benchmark	Measure	WOA-V-ET	BGOA	bGWO	BGSA	BPSO	bALO	BBA	BSSA
Data1	AVG	**141.3283**	378.4813	146.4914	541.6860	460.2917	1343.3640	425.2024	1393.7636
	STD	20.9900	20.0269	33.7014	24.8681	11.8739	188.1298	29.9367	47.1594
Data2	AVG	158.8498	371.4675	**147.3034**	511.1896	415.3392	1074.9705	419.9024	1297.9845
	STD	28.7516	13.3798	30.0253	16.7741	10.6119	114.1334	25.1458	28.8753
Data3	AVG	179.3935	388.6952	**158.0929**	515.2675	443.7511	1181.5459	419.8148	1317.7268
	STD	25.3290	19.6799	37.2793	24.9507	9.3245	173.0044	23.4784	29.2337
Data4	AVG	169.7138	350.0157	**143.8717**	514.3160	432.3967	1119.8705	427.9225	1316.8528
	STD	25.5382	11.2023	27.8197	22.8927	8.7900	145.2760	33.2796	27.9399
Data5	AVG	174.8223	366.0586	**157.1096**	519.8671	434.1215	1167.9915	428.8212	1320.0643
	STD	24.2564	17.5830	39.7709	16.3028	9.8424	180.7015	19.4970	31.8269
Overall Ranking	F-Test	1.8	3	**1.2**	6	4.8	7	4.2	8

Table A.29: Comparison between WOA-V-ET and other optimizers based on the average number of features

Benchmark	Measure	WOA-V-ET	BGOA	bGWO	BGSA	BPSO	bALO	BBA	BSSA
Data1	AVG	**2.2000**	27.3000	9.1000	52.8667	47.6333	75.0000	45.8000	63.0667
	STD	0.7611	7.8131	3.4576	5.0701	5.4487	12.7144	7.4436	5.5766
Data2	AVG	**3.5333**	25.8667	9.0333	53.6333	39.3333	59.7667	47.2333	56.7333
	STD	2.6876	9.0544	2.6061	4.9024	5.6038	10.8681	5.5191	4.1267
Data3	AVG	**2.6667**	28.5667	10.1667	52.9333	46.6667	69.7333	43.1667	62.1333
	STD	1.5388	9.2836	3.9661	4.4251	4.2858	13.5340	8.6785	5.7878
Data4	AVG	**2.5333**	21.2333	8.8000	52.6333	41.0000	65.5000	47.6667	60.0667
	STD	1.4320	6.5794	1.9191	6.6461	5.2850	14.8968	6.0077	5.0099
Data5	AVG	**4.4667**	26.0667	10.1667	54.0000	44.1000	72.3667	44.2667	62.2333
	STD	3.4415	7.1820	4.4263	4.4256	5.8566	14.6605	7.0169	6.2239
Overall Ranking	F-Test	**1**	3	2	6	4.4	8	4.6	7

Table A.30: Comparison between WOA-V-ET and other optimizers based on average fitness

Benchmark	Measure	WOA-V-ET	BGOA	bGWO	BGSA	BPSO	bALO	BBA	BSSA
Data1	AVG	**0.0003**	0.0035	0.0313	0.0846	0.0101	0.0796	0.0758	0.0585
	STD	0.0004	0.0009	0.0349	0.0220	0.0075	0.0149	0.0368	0.0259
Data2	AVG	**0.0025**	0.0121	0.0146	0.0243	0.0164	0.0196	0.0206	0.0192
	STD	0.0042	0.0053	0.0025	0.0245	0.0032	0.0010	0.0092	0.0004
Data3	AVG	**0.0061**	0.0754	0.0895	0.1121	0.0844	0.0973	0.1153	0.0929
	STD	0.0171	0.0252	0.0054	0.0248	0.0042	0.0026	0.0390	0.0050
Data4	AVG	**0.0049**	0.0635	0.0808	0.0986	0.0828	0.0860	0.1045	0.0851
	STD	0.0171	0.0323	0.0005	0.0225	0.0006	0.0013	0.0218	0.0006
Data5	AVG	**0.0113**	0.0763	0.0862	0.1021	0.0870	0.0939	0.1020	0.0912
	STD	0.0138	0.0217	0.0100	0.0295	0.0048	0.0022	0.0165	0.0032
Overall Ranking	F-Test	**1**	2	3.4	7.6	3.6	6.2	7.2	5