The Journal of Supercomputing (2024) 81:254
https://doi.org/10.1007/s11227-024-06677-7

A hybrid machine learning approach for feature selection in designing intrusion detection systems (IDS) model for distributed computing networks

Yashar Pourardebil Khah[1] · Mirsaeid Hosseini Shirvani[1] · Homayun Motameni[1]

Accepted: 31 October 2024 / Published online: 7 December 2024

Abstract

This paper presents a hybrid Machine Learning (ML)-based feature selection algorithm to create an Intrusion Detection Systems (IDS) model in distributed computing environments such as in Internet of Things (IoT) requests. IoT applications need frequent network connections to reach either the cloud or fog resources. Since reliability and trust are very prominent attributes in distributed systems, utilizing the ML-based algorithm is vital to efficiently detect malicious/attack behaviors at the earliest time with high accuracy. To address the issue, a hybrid ML-based algorithm is presented that includes four phases. In the first phase, the pre-processing including cleansing and normalization on the relevant dataset is performed. Then, the two next phases, namely heuristic-based and meta-heuristic-based approaches, are passed for the training step. In the second phase, three effective filter-based heuristic feature rankers are utilized to sort features according to their importance. Afterward, the fuzzy TOPSIS approach is also applied to prepare a consensus among them returning top-K features. In the third phase, to optimize the consensus feature subsets, a novel Discrete Gray Wolf Optimization Algorithm (DGWA) as a meta-heuristic approach is designed. It leads to a balance in exploitation and exploration searching areas. The effectiveness of the proposed hybrid model is tested in the fourth phase on the famous NSL-KDD and UNWS-NB15 datasets against some state of the art. The simulation results of running different scenarios prove that the proposed hybrid ML-based model gives an average 10.60%, 15.85%, 3.30%, 4.39%, and 2.03% improvement in testing datasets in terms of *accuracy*, *precision*, *recall*, *F-cost*, and *specificity* against other comparative state of the art in the same conditions.

Keywords Intrusion detection systems (IDS) · Feature selection · Discrete gray wolf algorithm (DGWA) · Fuzzy TOPSIS

✉ Mirsaeid Hosseini Shirvani
mirsaeid_hosseini@iau.ac.ir

[1] Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

1 Introduction

With the advent of new technologies and their advancement, the world has become connected and smart. The IoT-based smart devices make the world easily connected and smarter. The IoT devices and gadgets require computing resources provided by either cloud or fog servers via frequent network connections [1, 2]. However, smart devices and gadgets are prone to nefarious, malicious, adverse abnormal or attack behaviors in the broad global networks [3]. One of the most famous threats is a Denial of Service (DoS) attack in which the authorized user is precluded from reaching available resources by the attacker [4]. Security, trust and reliability are prominent attributes that must be supplied by distributed networks [4]. In these open systems, Intrusion Detection Systems (IDS) can detect malicious behaviors by comparing the coming network traffic with other previously discovered patterns from logged data at the earliest; then, the predefined countermeasures are performed if necessary. A Network IDS (NIDS) monitors network traffic to discover potential threats and attacks [3]. NIDSs are categorized into signature-based and anomaly-based network traffic. To detect malicious behavior, a signature-based approach matches coming traffic to one of the famous attacks. On the other hand, the anomaly-based approach creates a normal profile from any normal treatment in the network. Then, any violation from this kind of profile is taken as a possible attack. However, signature-based NIDS may fail to detect unknown attacks; therefore, anomaly NIDS is recommended. In this line, applying Machine Learning (ML) approaches is beneficial to detect each kind of network attack timely with high performance. The experienced network connections and network traffic in the past are logged in the network repository. The labeled connection whether is normal or an attack has been also recorded. The goal of ML-based approaches is to design a classification model from logged patterns to detect malicious behavior at the earliest time. Then, the incoming traffic can be recognized whether is safe or unsafe by applying the proposed classification model. The hurdle part in the path is to encounter with dimensionality curse. Several feature selection algorithms have been presented in the literature to reduce feature dimensionality and omit irrelevant and additional features. All of them suffer from the "curse of dimensionality" and huge search space in which designing an efficient feature selection algorithm that returns the optimal and condensed subset of the most effective features is crucial [5]. This problem computationally belongs to NP-Hard problems [6]. So, presenting an efficient solution is favorable. Miscellaneous heuristic-based and meta-heuristic-based approaches have been suggested in the literature to solve feature selection problems. Filter-based and wrapper-based are the most famous approaches and the hybrid approach is also added to enhance their worth. On the other hand, several meta-heuristic algorithms such as Genetic Algorithm (GA) [7–9], Crow Search Algorithm (CSA) [10, 11], and Gray Wolf Algorithm (GWA) [12–14] have been extended to figure out feature selection problems. However, for the sake of the special case of cloud/fog/IoT context and the high dimensionality of used datasets leads to face this complex problem. The IoT (machine, human, or relevant gadget)

continuously connects to the network for reaching to the cloud, fog, or other servers for requested resources. Then, the network traffic includes different patterns. So, the IDS model should extract patterns to create a detection classifier [15]. To this end, feature selection is a very important phase that guarantees the model's speed and accuracy [16, 17]. To solve this complex problem, a hybrid algorithm is presented. To verify the proposed model, two well-known datasets are incorporated. In this line, one of the most utilized datasets, i.e., NSL-KDD is selected [18, 19]. After preprocessing, the main dataset is split into two datasets NSL-KDD-train (80%) and NSL-KDD-test (20%). To create the classification model, the NSL-KDD-train dataset is used. To test the model, the NSL-KDD-test dataset is considered as the newcomer traffic in which the proposed model should detect whether incoming traffic is normal or not with a high degree of accuracy. In addition, another famous dataset named UNSW-NB15 is utilized to evaluate the effectiveness of the proposed hybrid classifier algorithm because it is more commensurate with the current low-footprint attacks [3, 20]. This dataset has more features that are possibly more effective in attack detection, especially in an intricate network. UNSW-NB15 is also partitioned into training and testing datasets. To solve this complex ML-based classification IDS model, this paper presents a hybrid four-phase method. In the first phase, the cleansing and normalization processes are done to prepare a normal and unbiased dataset. Also, some categorical features are converted to meaningful numerical features. Since the datatypes of the majority of features are numerical, three effective filter-based feature rankers among abundant rankers in the literature are selected. The selected filter-based rankers are Chi-square [6, 16], Pearson correlation coefficient [6, 16], and mean absolute deviation (MAD) [16] each of which assigns a rank value to each feature in the dataset. To use a better subset of features in the proposed model, the consensus approach needs to be applied. To this end, a fuzzy Technique for Order of Preference by Similarity to the Ideal Solution (TOPSIS) as a multi-criteria decision-maker is applied to aggregate the output of all rankers [21]. The fuzzy TOPSIS returns a set of top-K features (K: is a user-defined parameter) among the full set of features; this process is done at the end of the second phase by incorporating the consensus on the output of three rankers. In the third phase, to optimize a set of top-K features for producing the possible denser subset of features, a novel discrete version of the gray wolf optimization algorithm (DGWA) is presented. The proposed DGWA mimics traditional GWA, but it adds some new discrete local and global operators commensurate with the discrete search space of the stated problem. Finally, the model is tested in the fourth phase by applying the NSL-KDD-test and UNSW-NB15-test datasets taken as new incoming traffic in the network. Then, the accuracy and all relevant evaluation metrics are measured for model assessment.

Therefore, the contributions of this paper are listed below:

- It utilizes three prominent filter-based feature rankers commensurate with the used datasets to prepare three different lists of feature rankings.
- It utilizes a fuzzy TOPSIS tool to return a subset including top-K features as a consensus of the output of different ranking features.

- To possibly optimize the top-K feature set, the novel discrete gray wolf optimization algorithm (DGWA) is customized with new exploration and exploitation operators for efficient permutation of the discrete search space.
- It verifies the performance of the proposed model on famous datasets against some state of the art in the same conditions.

The rest of the paper is structured as follows. Section 2 gives a related work. The system framework is presented in Sect. 3. Some preliminary concepts are introduced in Sect. 4. Section 5 presents the main hybrid algorithm which solves the feature selection problem. The proposed algorithm is assessed in Sect. 6. Section 7 concludes the paper along with future directions.

2 Related work

This section reviews ML-based IDS networks and feature selection approaches in this context. ML-based approaches have different applications in a variety of industries such as in healthcare systems which analyze the utilized IoT devices for further processing and control [22], or in the financial industry to detect fraud detections [23], or in fog and cloud industries efficiently allocates available resources to IoT applications which request resource [24], etc. The investigation reveals that the literature can be classified into heuristic-based including filter-based and wrapper approaches; meta-heuristic-based, and also hybrid methods [25].

A filter-based heuristic algorithms were presented by Ayad et al. [1] to create an anomaly detection system in the IoT context. The used datasets were BoT-IoT [26] and ToN-IoT [27]. To create the attack detection model, the authors utilized the Pearson correlation coefficient (PCC) [6], Spearman CC (SCC) [28], and Relief [16] ranking methods for sorting the features based on their importance for classification. Every applied filter approach assigns a priority to a feature of a given dataset; then, the main recursive elimination heuristic is applied to return the final optimal subset of features. In the real world, optimization is a very important issue to reduce costs or increase benefits. For instance, clustering and optimization approaches are extended for power management in distributed systems [29, 30]. A Kalman filter as an ML-based approach was performed to estimate and predict low-load servers suggested for users [31]. An optimal heuristic feature selection algorithm was proposed by Siddiqi and Pak to present ML-based IDS [5]. They used CIC-IDS 2017 and ISCX-IDS 2012 datasets and extended their previous promising filter approach by incorporation of Yeo-Johnson and Pearson correlation [32]. In fact, it omits the same performance features from the full set of attributes. Kundu and Mallipeddi utilized a bunch of filter-based rankers on 18 different datasets to have the best feature selection [16]. Then, they formulated the feature selection problem into a multi-objective optimization issue. The authors customized the NSGAII algorithm to select a subset of efficient features for their proposed model. A cyber attack detection model for threat detection in IoT-based networks was presented by Kumar et al. by using a feature reduction heuristic [15]. This model uses PCC, gain ration, and random forest mean decrease accuracy (RFMDA) heuristics to create an attack detector

classifier for the IoT context. Afterward, the testing on NSL-KDD, BoT-IoT, and DS2OS datasets proves the effectiveness of the proposed classifier model. Zouhri and Ratnani examined five univariates filter-based feature selections such as relief, Pearson correlation, mutual information, ANOVA, and Chi2 on the CIC-IDS2017, CSE-CIC-IDS2018 and CIC-ToN-IoT intrusion detection datasets [33]. The simulation results of their experiments prove how utilizing the XGBoost and random forest trained with multivariate methods, such as CON and DISR, can shorten the size of used features without any side effect on the classification performance and its accuracy. A wrapper-based feature selection algorithm was suggested by Al-Yaseen et al. for preparing IDS in IoT networks [34]. This method utilizes a differential evaluation algorithm to excerpt the most efficient features where ML-classifier is exploited after selecting the feature subset for testing scenarios on NSL-KDD dataset. A hybrid wrapper sequential feature selection method is applied by using an optimized extreme learning machine and support vector machine (SVM)-classifier to present an accurate attack detector model in computer networks [8]. A genetic-based feature selection and classifier was proposed by Halim et al. [7]. To conduct the solutions toward promising districts, they introduced a new fitness function which considers data similarity. This classification GA-based algorithm classifies network traffic into normal and attack classes. The examinations have been done on three network traffic datasets: CIRA-CIC-DOHBrw- 2020, UNSW-NB15, and BoT-IoT. A modified gray wolf optimization algorithm was extended for creating an IDS model in distributed networks [14]. To improve the quality, the initial swarm of wolves is produced by incorporating the information gain (IG) calculation on all features. Also, the defined fitness function is a function of the false-positive rate, the false-negative rate, and the ratio of the number of used features to all features that should be minimized. A meta-heuristic ML algorithm was proposed by Sanju to design an IDS classifier model [2]. He utilized ensemble deep learning of recurrent neural networks to reduce the model's error. In the middle of the work for feature selection, the Harris hawk optimization algorithm is applied by elite selection strategy. Keserwani et al. proposed a mixed GWO algorithm with crow search algorithm (CSA) to prepare an IDS classifier model in the cloud computing environment [12]. The optimal classifier was tested on three well-known available datasets: NSL-KDD, UNSW-NB15, and CIC-IDS 2017. A CSA as an efficient meta-heuristic algorithm was proposed by SaiSindhuTheja and Shyam for DoS attack detection in a cloud computing environment [10]. The proposed hybrid algorithm has two stages. The features of a given dataset are bounded by using oppositional CSA (OCSA) and a recurrent neural network (RNN) classifier is applied to create attack detector model. In the testing phase, the testing dataset is fed to the proposed model for measurement of the performance evaluation metrics. A hybrid meta-heuristic ensemble triple filter rankings, such as Chi-square, mutual information (MI), and PCC was presented and aggregated them by incorporation of Pareto dominance concepts to create an IDS detector classifier [9].

Table 1 reviews related works in terms of the category of algorithm, applied approach, the used datasets, evaluation metrics, and the limitation of proposals.

This investigation reveals that abundant filter-based approaches are being used for solving feature selection problems because of their speed; on the other hand, the

Table 1 The literature review with comparison viewpoint

Author(s)/Ref./year	Category	Applied approach	Utilized dataset (s)	Evaluation metrics	Limitation
Ayad et al. (2024) [1]	Heuristic (filter-based)	PCC, Spearman CC(SCC), and relief	BoT-IoT and ToN-IoT	Accuracy, recall, precision, f-measure, and duration time	This is a fast approach, but for high-dimensional datasets it needs further processing
Siddiqi and Pak (2020) [5]	Heuristic (Filter-based)	Yeo-Johnson and Pearson correlation	CIC-IDS 2017 and ISCX-IDS 2012 datasets	Accuracy, recall, precision, and f1-score	Since it utilizes a single filter-based feature selection, the results need to be more accurate
Kundu and Mallipeddi (2022) [16]	Heuristic (Filter-based)	A bunch of feature rankers such as PCC, Chi-square, relief, mean absolute deviation (MAD), etc.	18 different datasets	Accuracy and Runtime	Some of them are fast, but low performance
Kumar et al. (2021) [15]	Heuristic (Filter-based)	PCC, Gain Ration, and Random Forest Mean Decrease Accuracy (RFMDA)	NSL-KDD, BoT-IoT, and DS2OS datasets	Accuracy, detection rate, precision, and f1-score	Applying this approach to larger datasets needs smarter and complicated filter-based methods
Zouhri and Ratnani (2024) [33]	Heuristic (filter-based)	Relief, Pearson correlation, mutual information, ANOVA, and Chi2	CIC-IDS2017, CSE-CIC-IDS2018 and CIC-ToN-IoT intrusion detection datasets	Accuracy, recall, precision, and f1-score	Mixing the presented approaches with meta-heuristics can enhance the performance provided the model is not time-limited
Al-Yaseen et al. (2022) [34]	Heuristic (wrapper-based)	differential evaluation algorithm to excerpt the most efficient features	NSL-KDD	Accuracy, detection rate, and false alarm rate	This procedure is rather slow in comparison with filter-based rankers
Maseno and Wang (2024) [8]	Heuristic (Wrapper-based)	optimized extreme learning machine (ELM) with an SVM (support vector machine) classifier	IoT_ToN and UNSWNB15	Accuracy, precision, and, recall	This kind of approach intrinsically suffers from heavy computations although it returns acceptable results

Table 1 (continued)

Author(s)/Ref./year	Category	Applied approach	Utilized dataset (s)	Evaluation metrics	Limitation
Sanju (2024) [2]	Meta-heuristic-based	Harris hawk optimization and mixing deep earning RNN	IoT-23, UNSW-NB15, and CIC-IDS2017	Accuracy, recall, precision, AUC_ROC, and f1-score	Application of this model faces time challenges in real-time systems
SaiSindhuTheja and Shyam (2021) [10]	Meta-heuristic-based	Oppositional crow search algorithm (OCSA)	KDD CUP 99	Accuracy, detection rate, precision, and f1-score	Since it utilizes a recursive neural network, it is time-consuming in real-time distributed systems
Halim et al. (2021) [7]	Meta-heuristic-based	Genetic algorithm	CIRA-CIC-DOHBrw-2020, UNSW-NB15, and Bot-IoT	Accuracy and Sensitivity	Incorporating filter-based ranking methods cab improve the quality of the proposed model
Alzaqebeh et al. (2022) [14]	Meta-heuristic-based	Gray wolf optimization algorithm (GWA)	UNSW-NB15	Accuracy, F-score, false error rate (FER), false-positive rate (FPR), and G-mean	Since it only uses information gain (IG), it can be improved from different angles by trying other filter-based approaches before calling modified GWA
Kumar et al. (2023) [9]	Hybrid meta-heuristic-based	Triple filter rankers: PCC, Chi-square, and mutual information mixing by genetic algorithm	Bot-IoT	Accuracy and number of selected features	The quality of performance is high, but it is rather slow for real-time scenarios
Keserwani et al. (2020) [12]	Hybrid meta-heuristic	Gray wolf optimization algorithm and CSA (IDS-GWA-CSA)	NSL-KDD, UNSW-NB15, and CIC-IDS	Accuracy, detection rate, precision, and f1-score	As it engages a semi-deep learning approach, it works rather slowly

wrapper approach is a little slow and has relatively high computations. The wrappers are usually commensurate with low-dimension datasets. For high-dimensional datasets, the hybrid approaches are recommended to shrink search space smartly; then, accurate searches are performed on the bounded search space by applying the most efficient meta-heuristic approaches this is the reason the current paper suggests a four-phase hybrid feature selection and classifier model.

3 System framework

CISCO presented three-layered distributed computing systems [35, 36]. Figure 1 depicts our proposed system framework derived from three-layered CISCO cloud-fog-IoT layers. In the IoT layer, users request to use computing resources whether placed in either cloud or fog platforms.

Based on the previous network connection requests in the history of users, the information of all connections is logged in the form of datasets. The normal and attack connections with their features have been saved in addition to their labels. This logged file has several undiscovered patterns for presenting a model for attack detection in requesting cloud and fog resources. So, this paper presents a classification model based on the logged data (relevant datasets) to detect normal or attack connections in real scenarios in which the system owner performs the countermeasures to preclude detriments. In the system framework, after creating the detection classifier model, the input network connection to reach cloud or fog resource via network connection can be recognized to be similar to normal or attack behavior. In case of attack detection, the countermeasure procedure is called if necessary. The proposed IDS model can be run either on cloud or fog servers depending on the time-limitation requests.

4 Preliminaries

Since this paper presents a model-driven solution, it should give a bright insight into the detector solution and what kind of data is being used. The machine learning (ML)-based IDS detector model utilizes datasets including different features. Every proposed model implicitly proves which of the features has a drastic impact on the final classifier model. In fact, there are several undiscovered patterns in which the proposed IDS model should recognize timely. To this end, a famous and prevalent being used dataset in different networks is introduced. As feature selection is an NP-hard problem, existing literature tries to find the most efficient and at the same time the condensed subset of features with the possible lowest number of applied features to reach the goal. Therefore, several ranker procedures in the form of filter-based approaches have been presented in the literature to give the best subset of effective features. However, each of these has its strengths and shortcomings. After investigation and experiments, three of the strongest rankers that are commensurate with the used dataset are selected in this paper to construct a cyberattack detector model in the training phase by using the training dataset. Also, to obviate this hurdle, a strong multi-criteria decision-maker

tool that is fuzzy TOPSIS is applied to aggregate for reaching a consensus of the results of rankers for reaching an ensemble decision. Till now, the Fuzzy TOPSIS returns the top-K features in which the parameter K is defined by the user. To reach the final concrete results, a novel subjective meta-heuristic algorithm is applied. In other words, a subjective discrete gray wolf optimization algorithm is applied to return the final subset of optimal selected features. Finally, the performance of the suggested model is verified by testing the datasets in the testing phase.

4.1 Dataset definition

To examine the proposed IDS model, two prevalent utilized datasets are engaged for IDS which are NSL-KDD [37] and UNSW-NB15 [3]. One of the most applicable datasets in network IDS is KDD99 [18]. It is being pervasively applied to all kinds of networks. However, IoT-based networks need ample data with more effective samples. To address the issue, the NSL-KDD dataset was created from KDD99 and prevalently used [18, 19]. It has frequent samples in comparison with KDD; also, it omits additional/repetitive records to preclude biased results. NSL-KDD is also split into two datasets, namely, NSL-KDD-train and NSL-KDD-test each of which has 41 features and one label (Totally 42 features). Among features 38 out of 42 are numerical either discrete or continuous; 3 are categorical, and the last one is nominal. The former dataset (NSL-KDD-train including 125,973 samples) is utilized for training in creating a classifier model whereas the latter (NSL-KDD-test including

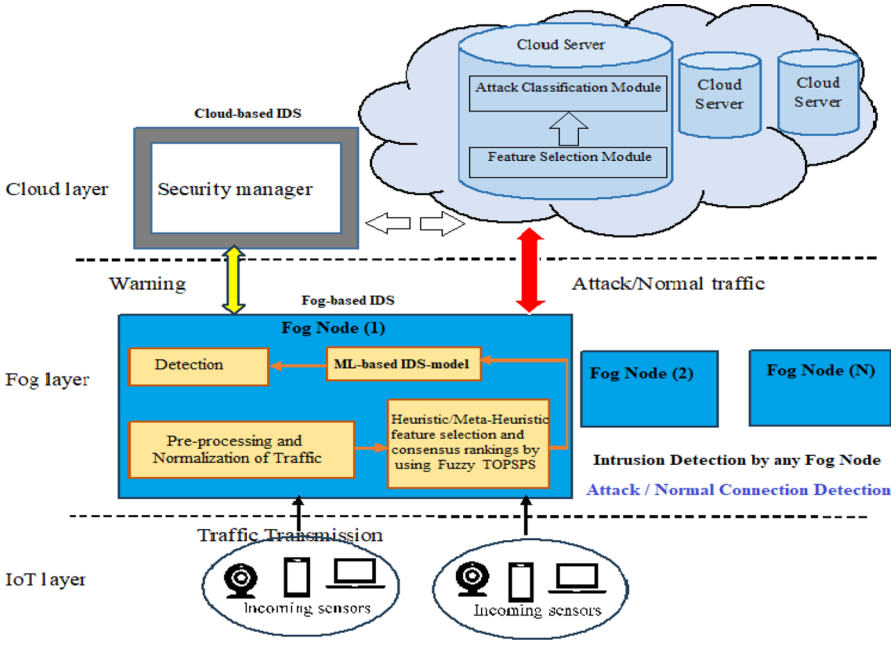


Fig. 1 The Proposed System Framework

22,544 samples) is applied for testing and performance evaluation of the generated model. Table 2 provides all features, their position number in a sample of the used NSL-KDD dataset, and associated datatypes.

In samples of NSL-KDD, the label feature (#42) includes 5 kinds of classes that are normal and 4 kinds of attacks. The categories of attacks are DoS, Probe, R2L, and U2R attacks. For the sake of simplicity, the label feature that is nominal is converted to a numerical data structure in the pre-processing phase. Respectively, values 0,1,2,3, and 4 are assigned to "normal" connection, "DoS," "Probe," "R2L," and "U2R" attacks. As mentioned earlier, different kinds of attacks in the NSL-KDD dataset are denial of service (DoS), Probe, R2L, and U2R which are:

- "DoS" is an attack class in which a legitimate user cannot reach available resources. Some of the relevant features are "syn flooding," "source bytes," and "percentage of packets with errors."
- "Probing" as another malicious behavior is to gain information about other remote victims such as a port scan attack. Associated features are "duration of connection" and "source bytes."
- "U2R" attack is unauthorized access to the local root to login into the victim system. The attacker tries to reach the root of admin privileges. The buffer overflow malicious behavior is a kind of U2R attack.
- "R2L" attack unauthorized access from a remote system. An attacker adversely logins into a remote system to gain the local root privileges; some of the relevant features are "network level features," "duration of connection," "service requested," "host level," and "number of failed login attempts."

However, every kind of real attack is provided in Table 3 for more elaboration. Table 3 tabulates all kinds of known attacks for each attack class of the NSL-KDD dataset.

In addition to Tables 2 and 3, Table 4 provides the number of items whether is normal or an attack in each class either in training or test datasets.

Table 2 The specification of NSL-Train and NSL-Test datasets [18, 19]

Data type	Feature name (position number of feature in the dataset)
Nominal (categorical)	Label (42), Protocol_type(2), Service(3), Flag(4)
Binary (0/1)	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21), is_guest_login(22)
Numeric	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23), srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29)
	diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36),
	dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

Table 3 Specification of all kinds of abnormal behaviors labeled to attack in the NSL-KDD dataset [18, 19]

Attack Category (attack Class)	Attack Name (Total Types)
DoS	Back, Land, Neptune, Pod, Smurf,Teardrop,Apache2, Udpstorm, Processtable, Worm (10)
Probing	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint (6)
R2L	Guess_Passwd, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named (16)
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps (7)

Table 4 Number of items in each normal/attack class for the NSL-KDD dataset [18, 19]

Class name	Number of training samples in each class	Number of training samples in each class
Normal	67,343	9711
DoS	45,927	7458
Probe	11,656	2421
U2R	52	200
R2L	995	2754
Number of Total Samples	125,973	22,544

As mentioned earlier, the UNSW-NB15 dataset is well-suited for modern low-footprint attacks. It included 9 different kinds of attacks such as DoS, generic, worms, backdoors, exploits, fuzzers, shellcode, and reconnaissance; in addition, it includes normal profiles. The mentioned dataset comprises 49 features. It has 175,341 and 82,332 samples, respectively, in its training and testing datasets.

Table 5 The specification of all kinds of abnormal behaviors labeled to attack in the UNSW-NB15 dataset [3]

Data type	Feature name (number of feature in dataset)
Binary (0/1)	is sm ips ports(36), is ftp login (39), Label (49)
Float	Dur(7), Sload (15), Dload (16), Sjit (27), Djit (28), Sintpkt (31), Dintpkt (32), tcprtt (33), synack (34), ackdat (35)
Integer	sport (2), dsport (4), sbytes (8), dbytes (9), sttl (10), dttl (11), sloss (12), dloss (13), Spkts (17), Dpkts (18), swin (19), dwin (20), stcpb (21), dtcpb (22), smeansz (23), dmeansz (24), trans_depth (25), res_bdy_len (26), ct_state_ttl (37), ct_w_http_mthd (38), ct ftp cmd (40), ct_srv_src (41), ct_srv_dst (42), ct_dst_ltm (43), ct_src_ltm (44), ct_src_dport_ltm (45), ct_dst_sport_ltm (46), ct_dst_src_ltm (47)
Nominal	Srcip (1), dstip (3), proto (5), state (6),14 service (14), attack_cat (48)
Timestamp	Stime (29), Ltime (30)

Table 5 introduces all features of samples in the UNSW-NB15 dataset along with their datatypes.

As mentioned earlier, different kinds of attacks in the UNSW-NB15 dataset are denial of service (DoS), analysis, backdoor, exploits, fuzzers, generic, reconnaissance, shellcode, and worms which are:

- "DoS" is an attack class in which a legitimate user is precluded from reaching available
- "Analysis" is a different kind of penetration including port scan, spam, and html files.
- "Fuzzers" is an attempt leading to a program or network suspended by input an irrelevant random fuzzy input
- "Backdoor" is an attempt at security principles are bypass reaching a server or data
- "Exploit" is an attack in which the attacker knows how to penetrate a piece of the program from the operating system's breaches
- "Generic" is an act against all block-ciphers without talking about the structure of the block cipher.
- "Reconnaissance" is a simulation-based attack for gathering information from the system
- "Shellcode" is a piece of code in a payload packet for abusing software vulnerabilities
- "worm" is an attacker in which it replicates itself to scatter into other systems. It usually utilizes a computer network to distribute the spread itself.

Table 6 provides the information associated with the number of all kinds of known attacks for each attack class of the UNSW-NB15 dataset.

Table 6 Number of items in each normal/attack class for the UNSW-NB15 dataset [3]

Class name	Number of Training samples in each class	Number of Training samples in each class
Normal	56,000	37,000
Analysis	2000	677
Backdoor	1746	583
DoS	12,264	4,089
Exploits	33,393	11,132
Fuzzers	18,184	6062
Generic	40,000	18,871
Reconnaissance	10,491	3496
Shellcode	133	378
Worms	130	44
Number of Total Samples	175,341	82,332

The objective of ML-based IDS is to generate a classifier model that engages the condensed and the most effective subset of available features. Then, it should classify coming traffic in the accurate class between either normal or attack categories.

4.2 Feature ranker heuristics

There are several ranker approaches in the ML literature. We engage three of the most effective filter-based ranking feature selection approaches that are tailored to the utilized dataset. As the majority of features in the utilized dataset are numerical, Chi-square, mutual congestion, and Pearson Correlation Coefficient (PCC) rankers are applied to list features based on their importance for the classification model. For instance, when the correlation coefficient of two features is near to one, utilizing both of them does not necessarily add any worth because applying each of them returns rather the same results. PCC can shorten the number of independent features. To increase the reliability of data integration in Industrial IoT (IIoT), a hybrid optimization algorithm and density correlation degree can enhance the reliability and trusting the gained results [38]. Consequently, the proposed hybrid classifier encounters a smaller search space that returns the results quickly. Note that, to avoid unbiased results, the normalization technique is performed on feature values; then, the heuristic filter-based rankers are applied.

Chi-square(χ^2): The χ^2 statistics examines the hypothesis of whether two features F_1 and F_2 are *independent* or not with the degree of freedom [6]. If the hypothesis can be refuted, the features have a correlation otherwise there is not any correction between them. The χ^2 statistics value is bigger, the importance is greater in ranking. The $\chi^2(A, B)$ statistics value for two features A and B are measured by Eq. (1). The metrics C and R are the numbers of distinct values in features A and B, respectively.

$$\chi^2(A,B) = \sum_{i=1}^{C}\sum_{j=1}^{R}\frac{\left(o_{ij} - e_{ij}\right)^2}{e_{ij}} \tag{1}$$

The notation o_{ij} is the observed frequency of the actual joint event (A_i, B_j) and e_{ij} are expected frequency that is obtained via Eq. (2).

$$e_{ij} = \frac{Count(a_i \in \mathrm{A}) \times \mathrm{Count}(b_j \in \mathrm{B})}{Number\ of\ data\ tuples(n)} \tag{2}$$

Mean Absolute Deviation (MAD): It as a statistics-based calculation that measures the absolute deviation value from a central positon of data (usually mean

value) [16]. If $a_i \epsilon A$ and $\overline{A}$ is the mean value of feature A, the MAD value for each numerical feature A is measured by Eq. (3). The bigger value for a feature indicates its importance in ranking.

$$MAD(A) = \frac{1}{m}\left[\sum_{j=1}^{m}\left|a_i - \overline{A}\right|\right] \tag{3}$$

Pearson Correlation Coefficient (PCC): The independence checking is done on the numeric feature (X) and label class (Y). The Correlation Coefficient (CC) similarity-based measurement is done to indicate the degree of importance in ranking a feature. The amount of a correlation is high, the importance of the feature is high and the ranking is low. The correlation values get in [-1..1] whereas near 1 indicates the variable X and Y strongly depend on each other; zero means no relation is there and -1 means counter-dependent. Pearson correlation coefficient value between feature X and label Y is calculated by Eq. (4).

$$PCC(X, Y) = \frac{Cov(X, Y)}{\sigma_X . \sigma_Y} \tag{4}$$

where $Cov(X, Y)$ is the covariance of variables X and Y. In addition, notations σ_X and σ_Y are the variance of variables X and Y, respectively [6, 16].

Since each filter-based ranker approach assigns a special weight to each feature as a rank number, the outputs of them are different. To have consensus among rankers, a strong multi-criteria decision-making tool such as fuzzy TOPSIS is utilized to extract a subset of features by incorporating three rankers' output.

4.3 Fuzzy TOPSIS (A heuristic multi-criteria problem solver)

Fuzzy TOPSIS stands for Technique for Order of Preference by Similarity to Ideal Solution; this approach was firstly introduced by Hwang and Yoon [39]. It gets d-dimensional alternatives as options and constitutes a positive ideal solution (A^+) and a negative ideal solution (A^-). In other words, an artificial positive solution (A^+) encapsulates all of the best attributes of alternatives and an artificial negative ideal solution (A^-) encapsulates all of the worst attributes of alternatives. Note that attributes are categorized into two concepts cost and benefit. To construct A^+ the biggest values of the benefit attribute and the lowest values of the cost attributes are selected; accordingly, the biggest values of the cost attribute and the lowest values of benefit attributes are selected to create A^-. Then, it utilizes a d-dimensional Euclidian distance. TOPSIS method considers alternatives in the R^d domain and tries to find an option that is the nearest to (A^+) and is the farthest from the ideal solution (A^-) at the same time. Figure 2 depicts a two-dimensional search space. It can be generalized to the n-dimensional search space.

The term $A_i (i = 1, \ldots, M)$ is an alternative; the terms S_i^+ and S_i^- are the distance of an alternative A_i to the positive ideal solution (A^+) and the negative ideal solution

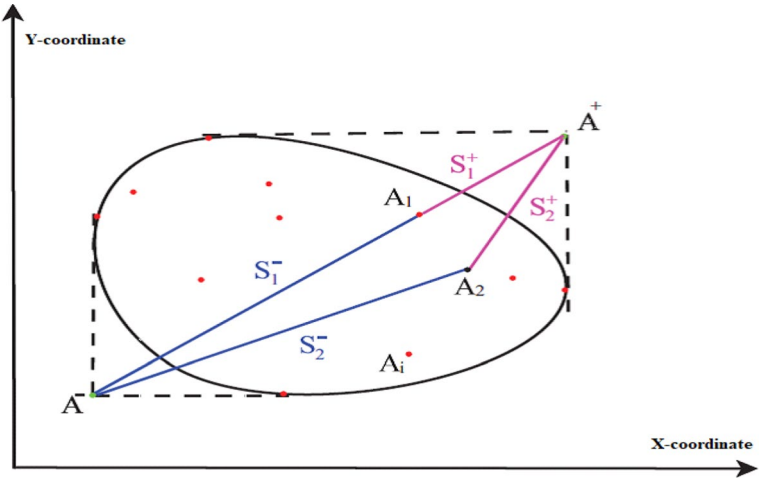


Fig. 2 A two-dimensional alternative in R^2 space

(A^-), respectively. Based on the measured distance, alternatives are ranked. How the fuzzy TOPSIS is customized at the end second phase of the proposed hybrid algorithm is elaborated in Sect. 5.

5 Proposed hybrid feature selection algorithm for intrusion detection system (IDS) in distributed networks

This section provides the proposed hybrid feature selection algorithm to create an efficient IDS classifier model in distributed networks. Finally, this proposed model can recognize the input connection network with the most accuracy. In other words, it classifies requested connection and data transfer into either normal or attack connection based on selected features in the model and it reacts in case of attack recognition with predefined countermeasures. Figure 3 illustrates the block diagram of the proposed hybrid algorithm.

The proposed framework includes 4 phases: preprocessing, feature selection (heuristic-based), training phase (meta-heuristic-based), and testing & evaluation. Algorithm 1 illustrated in Fig. 4 is designed to provide the details of the proposed hybrid approach.

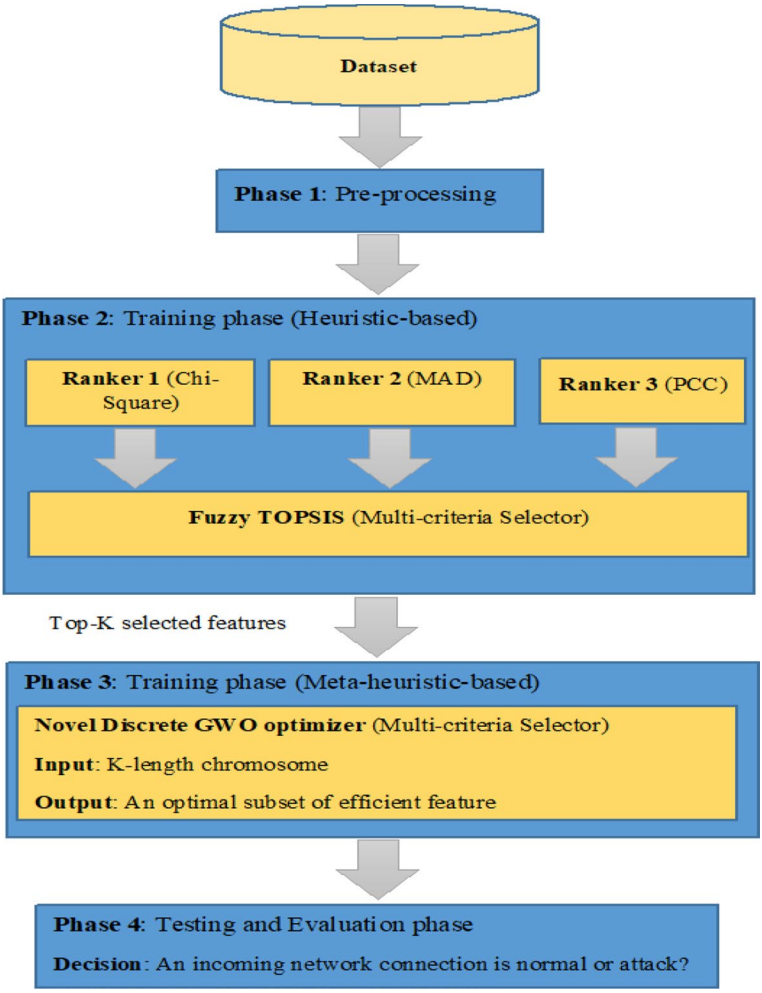


Fig. 3 The proposed four-phase ML-based feature selection algorithm

Fig. 4 Pseudo code presentation of proposed Algorithm 1

f_1	f_2	...	f_K
0/1	0/1	...	0/1

Algorithm 1 A Hybrid ML-based IDS for IoT networks.

Input
K: user-defined parameter;
/* K: is the length of chromosome
$D_{m\times N}$: A given an original dataset
$DTR_{m1\times N}$: A split training dataset
$DTS_{m2\times N}$: A split testing dataset
/* $m1$: is number of samples in training dataset
/* $m2$: is number of samples in testing dataset
/* N: is number of whole features */
SwarmSize : number of wolves in a swarm of solutions
r: number of filter-baser rankers; /* Here r=3 */
w: array [1.. r] of coefficient for ranker's importance; /* $w[1]+ w[2] +\ldots+ w[r]=1$, $0< w[i]<1$ */
WL [1..$SwarmSize$] : Swarm of encoded wolves
/* each wolf $WL[i]=[w_{i1}, w_{i2},\ldots, w_{iK}]$ is a K-length binary encoded */
MaxIter : maximum number of iterations
Output
An optimal subset of effective optimal features;

----------/* **Phase 1 for Pre-processing**/----------
1: **Perform** Pre-processing ($D_{m\times N}$)
2: **Split** normalized ($D_{m\times N}$) into $DTR_{m1\times N}$ and $DTS_{m2\times N}$ datasets respectively for training and testing ;
----------/* **Phase 2 for training by using Heuristic Filter Rankers**/----------
3: **Perform** Chi-Square-Ranking ($DTR_{m1\times N}$)
4: Let $Ranked_{Chi-Square} = \{CV_1, CV_2, \ldots, CV_N\}$ by using Eq. (1);
5: **Perform** MAD-Ranking ($DTR_{m1\times N}$)
6: Let $Ranked_{MAD} = \{MV_1, MV_2, \ldots, MV_N\}$ by using Eq. (3);
7: **Perform** PCC-Ranking ($DTR_{m1\times N}$)
8: Let $Ranked_{PCC} = \{PC_1, PC_2, \ldots, PC_N\}$ by using Eq. (4);
9: Let $K = 20$; /* K: for number of features */
10: Let r=3; /* r: number of filter-based rankers */
11: Let $w[1]= w[2] =\ldots= w[r]=1.0/ r$; /* Here, r=0.333 */
12: Let *TopK_Features* ← Call Fuzzy_TOPSIS($Ranked_{Chi-Square}$, $Ranked_{MAD}$, $Ranked_{PCC}$, K, $w[1..r]$);
----------/* **Phase 3 for training by using optimal Meta-Heuristic Ranker** */----------
13: Let *OptimalFeatures* ← **Call** DGWA-Filter (*TopK_Features*, $DTR_{m1\times N}$) by **Algorithm 2**;
----------/* **Phase 4 for testing**/----------
14: [accuracy, recall, precision, F-score] ← Call **Classifier** (*OptimalFeatures*, $DTS_{m2\times N}$)
15: return [accuracy, recall, precision, F-score];
16: **End** {**Algorithm 1**}

Algorithm 1 gets the dataset and related parameters as an input; then, it returns the subset of optimal features out of all features. All of the phases in the proposed Algorithm 1 are elaborated. As can be seen in Fig. 4, it performs pre-processing in the first phase.

First phase: Pre-processing

In this phase, the input NSL-KDD.csv file or other given dataset is copied in matrix$D_{m\times N}$. Additional and dummy records are omitted. For the sake of simplicity, the label feature in NSL_KDD that is nominal is converted to a numerical data structure in the pre-processing phase. Respectively, values 0,1,2,3, and 4 are assigned to "normal" connection, "DoS," "Probe," "R2L," and "U2R" attacks. Similar to Ref. [34], the nominal (categorical) feature can be converted into numerical without any side effect. For instance, the second feature in NSL-KDD is "Protocol_Type" that takes {"tcp," "udp," "icmp"} values. Therefore, a recommendation is to assign 0, 1, and 2 numeric values to "tcp," "udp," and "icmp" protocols in a network connection. Then, available features'

values are normalized by Eq. (5). After normalization of NSL-KDD, the dataset is split into 80% and 20% samples datasets that are NSL-KDD-training.csv and NSL-KDD-testing.csv files; these files have been opened in our python programming language, respectively, copied in 2-dim matrices $DTR_{m1\times N}$ and$DTS_{m2\times N}$. The former has 125,973 samples (records) and the latter has 22,544 samples (records). Accordingly, the same pre-processing procedure is done on UNSW-NB15. The numbers of samples in training and testing are 175,341 and 82,332 records in the UNSW-NB15 dataset, respectively. We pre-processed them by using *Pandas* and *DataFrame* capabilities in a Python programming environment [40]. Additional data has been removed to have unbiased processing. Since UNSW-NB15 includes a 10-label class, the label feature that is nominal (categorical) is converted to numerical values. Integer numbers in {0,1,…,9} assigned to "normal" connection, "DoS," "Analysis," "Backdoor," "Exploits," "Fuzzers," "Generic," "Reconnaissance," "Shellcode," and "Worms" attacks, respectively. In addition, to have a good distribution of values in different numerical features, the normalization technique is performed; there are several techniques in literature such as z-score [6], *min–max*-normalization [6], etc. To this end, *min–max*-normalization is performed on each numerical feature. If a feature A has values:Val_1,Val_2, …,Val_T; and the terms min_{Val} and Max_{Val} be the minimum and maximum values in feature$A$. Then, the new normalized value $Val\prime_i$ of Val_i is obtained by incorporation Eq. (5).

$$Val_i\prime = \frac{Val_i - min_{Val}}{Max_{Val} - min_{Val}} \tag{5}$$

Note that the new normalized values preserve the same relationship between the original data.

At the second phase, the feature selection approach is performed three times in lines 3, 5, and 7 of Algorithm 1, each of them returns ranked values for each feature in a given dataset. Afterward, a consensus function is done by calling the fuzzy TOPSIS approach in line 12.

Second phase: Feature Selection (Heuristic-based Training)

There are several filter-based numerical and frequency–based feature selection algorithms which rank the features according to their importance for classification. After examining the majority of these heuristic rankers, the three most effective among them have been excerpted which have returned better results. The rankers are Chi-square [6, 16], Mean Absolute Deviation (MAD) [16], and Pearson Correlation Coefficient (PCC) [6, 16] each of them assigns a rank to a feature. In the Chi-square method, the highest value of a feature (variable) means the higher importance (the lowest ranking). In MAD, a similar behavior is performed The mentioned filter rankers are called in lines 3, 5, and 7, respectively. The produced rank values are assigned in respective lists in lines 4, 6, and 8 of Algorithm 1. To reach a consensus result, the fuzzy TOPSIS tool is applied to encapsulate all of the outputs in the condensed final results; the result is a top-K subset of features. The fuzzy TOPSIS engine gets three ranking lists of features; then, it returns the final ranked list of features by incorporating the output of all rankers. Here, we explain how this phase works. Take a training dataset is a matrix $DTR_{m1\times N}$

including $m1$ number of samples each of which has N features:$F = \{f_1, f_2, \ldots, f_N\}$. After the execution of Chi-square, MAD, and PCC rankers, three lists of ranked features have been returned by them. Equation (6), Eq. (7), and Eq. (8), respectively, show the ranked features by Chi-square, MAD, and PCC heuristic filter-based rankers. The notations CV_i, MV_i, and PV_i are the values assigned to the i-th feature by calling Chi-square, MAD, and PCC filter-based rankers. Note that, the values of Chi-square, PCC, and MAD are taken as benefits because the bigger value is worthy [41].

$$Ranked_{Chi-Square} = \{CV_1, CV_2, \ldots, CV_N\} \tag{6}$$

$$Ranked_{MAD} = \{MV_1, MV_2, \ldots, MV_N\} \tag{7}$$

$$Ranked_{PCC} = \{PV_1, PV_2, \ldots, PV_N\} \tag{8}$$

Generally, the fuzzy TOPSIS receives r number of applied filter-based rankers each of which returned a list of weighted features (In this paper, r =3). Alternative matrix $A_{N\times r}$ is constituted in the first step. Since fuzzy TOPSIS should return top-K features, the alternatives are features that must be selected in the final subset. Therefore, alternative matrix $A_{N\times r}$ is created as can be seen in (9); then, K out of N is excerpted. The term K is a user-defined parameter. This process is named the "feature selection" phase.

$$A_{N\times r} = \begin{bmatrix} a_{11} & \cdots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{Nr} \end{bmatrix} \tag{9}$$

For instance, in our paper, $A_{41\times 3}$ for NSL_KDD dataset is created with N alternatives in $A=(A_1, A_2, \ldots, A_N)$ and r criteria in $C=(C_1, C_2, \ldots, C_r)$. The features are alternatives and the ranker values are criteria. Here, we have three criteria that can be seen in (10).

$$A_{41\times 3} = \begin{bmatrix} Chi. & MAD & PCC \\ CV_1 & MV_1 & PC_1 \\ CV_2 & CV_2 & PC_2 \\ CV_N & MV_N & PC_N \end{bmatrix} \tag{10}$$

To indicate the importance of the criteria, the weight is taken into consideration. So, the weight vector $\omega = [\omega_1, \omega_2, \ldots, \omega_r]$ is created in which $0<\omega_j<1$ and $\sum_{j=1}^{r} \omega_j = 1$. The weighted alternative matrix ($WA_{N\times r} = [wa_{ij}]_{N\times r}$) is calculated by considering the weight of every criterion. In fact,$WA_{N\times r}$ is calculated by multiplying the alternative matrix $A_{N\times r}$ by diagonal weight vector $\omega_{r\times r}$ in the alternative matrix by Eq. (11).

$$WA_{N\times r} = A_{N\times r} \times \omega_{r\times r} \tag{11}$$

For now, we take the same importance for every ranker; in other words, we take$\omega_1 = \omega_2 = \omega_3 = 0.333$. In the next step, a normalized decision relative matrix $DR_{N\times r} = [dr_{ij}]_{N\times r}$ is measured from $WA_{N\times r}$=$[wa_{ij}]_{N\times r}$ by Eq. (12) (for instance for NSL_KDD) where $i=1,\ldots,N$ and $j=1,\ldots,r$.

$$DR_{41\times3} = \begin{bmatrix} Chi. & MAD & PCC \\ dr_{1,1} & dr_{1,2} & dr_{1,3} \\ dr_{2,1} & dr_{2,2} & dr_{2,3} \\ dr_{43,1} & dr_{43,2} & dr_{43,3} \end{bmatrix} \tag{12}$$

A normalized value for each item dr_{ij} is calculated by Eq. (13).

$$dr_{ij} = \frac{wa_{ij}}{\sqrt{\sum_{i=1}^{N} wa_{ij}^2}} \tag{13}$$

In the next step, two r-dimensional ideal vectors are created which are positive ideal solution $A^+ = [x_1^+, x_2^+, \ldots, x_r^+]$ and negative ideal solution $A^- = [x_1^-, x_2^-, \ldots, x_r^-]$. Note that, x_j^+ in A^+ takes its value from the best value of j-th column in matrix DR and x_j^- in A^- takes its value from the worst value of j-th column in matrix DR. Since the first two columns are benefits, the biggest values are selected for corresponding features in A^+ and the smallest value is selected for the third which is the cost column. Accordingly, the smallest values are selected for corresponding features in A^- and the biggest value is selected for the third which is the cost column.

In the next step, the Euclidian distance between each alternative and positive and negative ideal solutions are measured by Eq. (14) and (15), respectively.

$$S_j^+ = \sqrt{\sum_{k=1}^{r} (x_k^+ - dr_{jk})^2}, j = 1, \ldots, N \tag{14}$$

$$S_j^- = \sqrt{\sum_{k=1}^{r} (x_k^- - dr_{jk})^2}, j = 1, \ldots, N \tag{15}$$

In the next step of the TOPSIS method, the closeness degree is calculated for each alternative by Eq. (16).

$$C_j = \frac{S_j^-}{S_j^+ + S_j^-}, j = 1, \ldots, N \tag{16}$$

In Eq. (16), the closeness value for each alternative is $0 < C_j < 1$. Then, sorting alternatives according to closeness value in decreasing order returns the real ranking best alternative to the worst ones. After utilizing the fuzzy TOPSIS, K out of N best ranking features are selected as a subset of the consensus best features. This subset can possibly have better performance by applying an efficient meta-heuristic algorithm in the next phase.

Third phase: Feature Selection (Meta-Heuristic-based Training)

This stage is one of the most important phases of the proposed hybrid algorithm which is the heart of it. It focuses on optimization. In fact, it receives an efficient subset

of consensus features. Then, it returns an optimal subset of features which is condensed and high-performance for using in the classification model. To this end, a novel discrete gray wolf optimization algorithm (DGWA) is presented. Since the canonical GWA is a continuous optimizer, we present a customized discrete version of the original GWA commensurate with the stated problem. GWA mimics the real gray wolf behavior. The herd of wolves follows democratic treatment. The wolves adopt a leader and his/her advisors. The rest wolves are followers. In the long term, the wolves cannot follow the old or weak wolf; in other words, they follow the strongest and most experienced wolf and their advisors. In artificial wolf optimization, the best wolf is named alpha, the second best is named beta, the third best is delta, and the rest that is followers are named omega. Since the top-K features have been selected from three ranking lists, the proposed DGWA uses K-length encoding for creating chromosomes as solution representatives. Take $F = \{f_1, f_2, \ldots, f_K\}$ be a top-K feature set that is the output of fuzzy TOPSIS. In this algorithm, the genes are binary. When the i-th feature is considered, i-th gene is taken "1" in a chromosome (a wolf or candidate solution) otherwise it takes "0." Figure 5 depicts a chromosome of a wolf which is a candidate solution.

Similar to several meta-heuristic-based algorithms, the number of random solutions (K-length wolves) are produced as a swarm of wolves (lines 1 through 5 in Algorithm 2). The algorithm is iterated until the termination conditions are satisfied. The wolf position changes are done by two strategies, i.e., local strategy (exploitation) and global strategy (exploration). For a local strategy, each wolf utilizes the knowledge of the experienced wolves (leader wolves) that are alpha, beta, and delta wolves. In fact, this wolf body chromosome is refined by the body chromosome of leader wolves. To this end, the voting behavior is done. For assigning each gene for a chromosome-like a wolf, the corresponding gene is taken from the consensus function of three genes associated with three leader wolves. It uses the majority consensus. For global strategy, a couple of exploration operators are designed each of which is randomly called to explore search space efficiently. To avoid from a biased behavior, the position changing of each wolf is randomly done either by exploitation or exploration. This casual behavior is done by drawing a real value $0 < q < 1$; if it is greater than 0.5, exploration is done (line 16 of Algorithm 2) otherwise exploration is done (line 22 of Algorithm 2). In the exploration phase one of the designed global search operators is called. To investigate the efficiency of each candidate solution (the effectiveness of a subset of features in a classifier), the fitness function is considered. In this paper, the

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
0	1	1	0	0	1	0	1	1	0

Fig. 5 A presentation of an encoded wolf as a candidate solution

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
0	1	0	1	0	0	1	1	1	0

Fig. 6 Pseudo code presentation of proposed Algorithm 2

fitness is the amount of *accuracy* of a solution as a prominent performance parameter that Eq. (17) measures. The proposed algorithm is repeated until the termination criteria are met; in this paper, a maximum number of iterations is experimentally set when the final solution has no drastic changes. Then, the best so far subset of features that returns the maximum accuracy is returned as a final solution. Figure 6 depicts the presentation of the main DGWA-based Feature-selection procedure in Algorithm 2.

Algorithm 2 DGWA-based Feature-selection Procedure.

Input
K: user-defined parameter;
/* K: is the length of chromosome
$D_{m\times K}$: A given dataset /*refined in columns*/
SwarmSize : number of wolves in a swarm of solutions
$WL\ [1..SwarmSize]$: Swarm of encoded wolves
/* each wolf $WL[i]=[w_{i1}, w_{i2},\ldots, w_{iK}]$ is a K-length binary encoded */
MaxIter : maximum number of iterations
Output
An optimal subset of effective features

/* initialize the swarm of wolves */
For $i \leftarrow 1$ to *SwarmSize* **Do**
  For $j \leftarrow 1$ to K Do
    $WL[i][j] \leftarrow$ random binary value
  End-For-*j*
End-For-*i*
For $i \leftarrow 1$ to *SwarmSize* **Do**
  $Fitness[i] \leftarrow accuracy\ (WL[i])$
End-For-*i*
Let leaders: WL_α, WL_β, and WL_δ be the first, second, and third best wolves in a swarm.
Let *BestSolution* $\leftarrow WL_\alpha$; BestFitness $\leftarrow Fitness(WL_\alpha)$
Iter $\leftarrow 1$
while Iter $\leq$ *MaxIter* **Do**
  For each wolf $WL[i]$ in Swarm where i =1,…, *SwarmSize* **do**
    Draw a random real $0< q <1$;
    if $q > 0.5$ **then**
    ----------(* Exploitation *)----------
      $TempW \leftarrow$ Call **Exploitation** $(WL[i], WL_\alpha, WL_\beta, WL_\delta)$ by running **Algorithm 3**
      if *Fitness* $(TempW)$ is better than *Fitness* $(WL[i])$ **Then**
        $WL[i] \leftarrow TempW$
        Fitness$(WL[i]) \leftarrow$ *Fitness* $(TempW)$
      end-if
      else
    ----------(* Exploration *)----------
        $TempW \leftarrow$ Call **Exploration** $(WL[i])$ by running **Algorithm 4**
        if *Fitness* $(TempW)$ is better than *Fitness* $(WL[i])$ **Then**
          $WL[i] \leftarrow TempW$
          Fitness$(WL[i]) \leftarrow$ *Fitness* $(TempW)$
        end-if
      end-if
    end-if
  end-for
  Update leader wolves: WL_α, WL_β, and WL_δ
  if *Fitness* (WL_α) is better than *Fitness* (*BestSolution*) **Then**
    BestSolution $\leftarrow WL_\alpha$
    Fitness(*BestSolution*) $\leftarrow$ *Fitness* (WL_α)
  end-if
  Iter $\leftarrow$ Iter +1;
end-while
return *BestSolution*
End {**Algorithm 2**}

In Algorithm 2, two phases are randomly performed to avoid biased behavior which are exploitation and exploration. In the exploitation phase, the position of under study wolf is changed by following the position of the leader wolves. Since our proposed gray wolf optimizer is a discrete version, the changes are done in a discrete manner. Algorithm 3 depicted in Fig. 7 is dedicated to performing exploitation. Its inputs are four K-length solutions, i.e., one of them is under study wolf, and the rest are leaders WL_{α},WL_{β}, and WL_{δ}. During the gene refinement, a kind of semi-voting manner is performed. To avoid early convergence, a random real $0 < q < 1$ is drawn; if it is greater than 0.5, the corresponding gene in the wolf is preserved otherwise the majority same gene from leaders is set in the corresponding gene of under study wolf.

Algorithm 3 Exploitation Procedure.

Input
K: user-defined parameter;
WL: a wolf under study;
/* K: is the length of chromosome
/* each wolf WL=[w_1, w_2,…, w_K] is a K-length binary encoded */
WL_{α}, WL_{β}, and WL_{δ}: leader wolves.
Output
An new changed wolf $NewWL$

1: **For** $i \leftarrow 1$ to K **do**
2: Draw a random real $0 < q < 1$;
3: **if** $q > 0.5$ **then** /* preserve the last position */
4: $NewWL[i] \leftarrow WL[i]$
5: **else**
2: $NewWL[i] \leftarrow$ Majority voting on genes of ($WL_{\alpha}[i]$, $WL_{\beta}[i]$, $WL_{\delta}[i]$);
3: **end-if**
3: **End-For**
4: return $NewWL[1..K]$
5: **End** {**Algorithm 3**}

To explore search space efficiently, the exploration phase is randomly stated by calling Algorithm 4. To have diverse exploring to observe more probable candidate solutions, diverse exploration behaviors are performed in Algorithm 4 illustrated in Fig. 8. It makes the search space permutated in an efficient way. Once the exploration trajectory is selected in line 22 of Algorithm 2, Algorithm 4 is called. It changes the current solution with one of five different exploration behaviors. To this end, one integer number is randomly drawn. If it is one, two different integer indices are drawn; afterward, the genes associated with two positions are exchanged. Then, the new wolf is returned; else if that a drawn integer is 2, the second exploration technique is done. In this way, the gene in an odd position is exchanged with the next even position; else if a drawn integer is 3, the third exploration technique is performed. Accordingly, the gene in the even position is exchanged with the next odd position; finally, if the drawn integer is 5, the last approach is done in which it draws an index. Then, it puts its not in the corresponding index. In other words, it puts "1" if it sees "0" otherwise it puts "0."

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
1	0	1	0	0	0	1	1	0	1

Fig. 7 Pseudo code presentation of proposed Algorithm 3

Algorithm 4 Exploration Procedure.

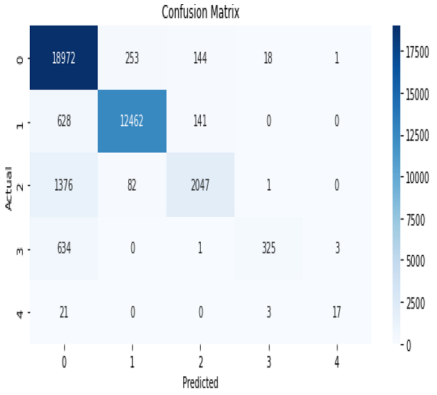
Input
K: user-defined parameter;
WL: a wolf under study;
/* K: is the length of chromosome
/* A wolf WL=[w_1, w_2,…, w_K] is a K-length binary encoded */
Output
An new changed wolf WL

1: Draw an integer number q in ~[1..5] ;
2: **if** q ==1 **then**
3: Draw two different integer numbers m and n in ~[1..K]
4: Exchange ($WL[m] \leftrightarrow WL[n]$);
5: **else if** q ==2 **then**
6: **For** $i \leftarrow$1 to $\lfloor K/2 \rfloor$ **do**
7: Exchange ($WL[2i-1] \leftrightarrow WL[2i]$);
8: **end-for**
9: **else if** q ==3 **then**
10: **For** $i \leftarrow$1 to $\lfloor K/2 \rfloor$ **do**
11: Exchange ($WL[2i] \leftrightarrow WL[2i+1]$);
12: **end-for**
13: **else if** q ==4 **then**
14: Draw two different integer numbers m and n in ~[1..K]; /* take $m < n$
15: $WL[1..m] \leftrightarrow WL[1..m]$);
16: $WL[m+1..n] \leftrightarrow$ Reverse ($WL[m+1..n]$);
17: $WL[n+1..K] \leftrightarrow WL[n+1..K]$);
18: **else**
19: Draw an integer numbers m in ~[1..K];
20: $WL[m] \leftarrow$ **not** ($WL[1..m]$);
21: **end-if**
22: return $WL[1..K]$
23: **End** {**Algorithm 4**}

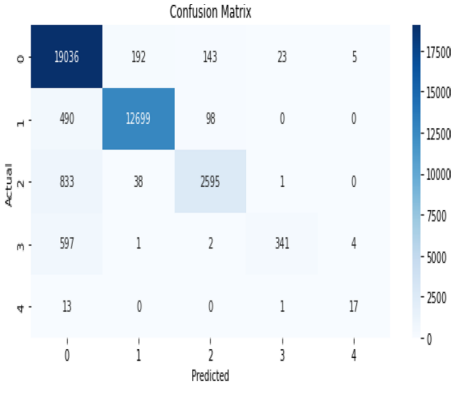
To show the application of Algorithm 4, take a 10-length wolf ($K = 10$) as Fig. 9 demonstrates. Once Algorithm 4 is called for exploration, take 4 is drawn in line 1. So, the condition in line 13 is true. Then, two integers 3 and 7 are drawn randomly. After executions of lines 15, 16, and 17, the new wolf depicted in Fig. 10 is returned.

Accordingly, if 2 is randomly generated in line 1 of Algorithm 4, the condition of line 5 is true. Then, the input wolf depicted in Fig. 9 is changed to the wolf depicted in Fig. 11.

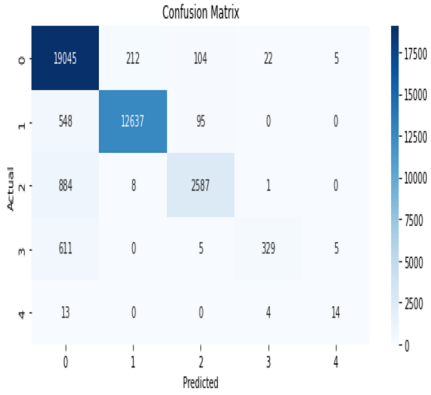
Note that, both exploration and exploitation changes never produce imperfect encoded solutions; therefore, it is not necessary to design Check&Correct (.) function [42].



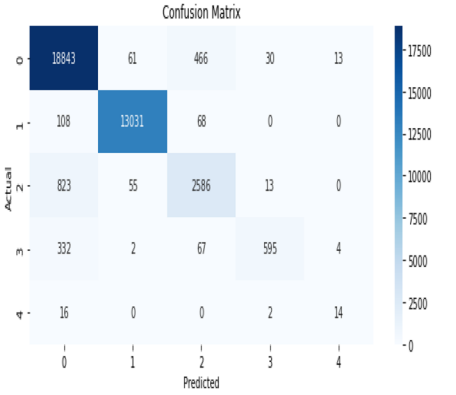
12) a. HF [13] gives 91% accuracy



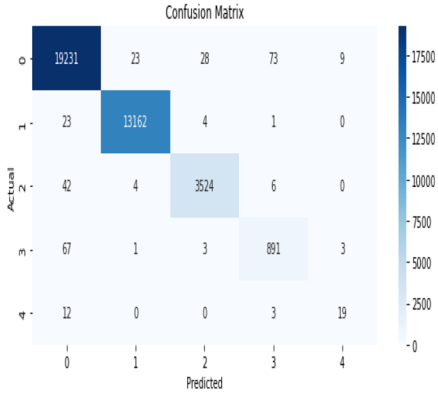
12) b. MCSA [10] gives 93% accuracy



12) c. NSGA-II [9] gives 93% accuracy



12) d. MGWA [32] gives 94% accuracy



12) e. HyDGWA gives 99% accuracy

Fig. 8 Pseudo code presentation of proposed Algorithm 4

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
0	1	1	0	0	1	0	1	1	0

Fig. 9 A sample encoded wolf

Fourth Phase: Testing & Evaluation phase

Now that the classification model is presented, the testing phase starts. The testing is done on testing datasets that are NSL-KDD-test.csv [43] and UNSW-NB15-test.csv files [44]. The detector IDS model was made according to the training data. Now, the performance of the detector model is verified according to the testing data. The performance evaluation is elaborated in Sect. 6.

Time Complexity of the Proposed Algorithm 1

A closer look at the statement of Algorithm 1 reveals that the time complexity depends on input parameters: K, m, N, *SwarmSize*, and *MaxIter*. Before measurement of the time complexity of Algorithm 1, the time complexities of Algorithm 2, Algorithm 3, and Algorithm 4 are presented. The time complexities of Algorithm 3 and Algorithm 4 are simply calculated by O(K) where $K < \mathrm{N}$. note that parameter N is the total number of features in the given dataset. In Algorithm 2, lines between 1 through 5 run a nested loop. It takes O($K \times SwarmSize$). Algorithm 2 runs while-loop *MaxIter* times. In the main loop, it performs either exploration or exploitation for every wolf in the swarm. Since either exploration or exploitation takes O(K), respectively, by calling Algorithm 2 and Algorithm 3, and there are *SwarmSize* wolves in a swarm, while-loop takes O($K \times SwarmSize \times MaxIter$). Overall, Algorithm 2 takes $K \times SwarmSize + K \times SwarmSize \times MaxIter$. The simplification leads to the time complexity of Algorithm 2 being O($K \times SwarmSize \times MaxIter$). To calculate the time complexity of Algorithm 1 needs investigation on all statements. In Algorithm 1, line 1 takes O(m). line 2 takes O(1) because of simple split. All callings in lines 3,5, and 7 take O(N) because all features are compared with the "label" feature to rank all features. The most effective statement is in line 13 that calls Algorithm 2. A closer look at them reveals that the time complexity of Algorithm 1 depends on the time complexity of Algorithm 2. Then, the time complexity of the proposed Algorithm 1 is O($K \times SwarmSize \times MaxIter$).

6 Performance evaluation

To verify the effectiveness of the proposed hybrid feature selection algorithm in detecting normal/abnormal network connections in distributed systems, different scenarios are conducted. To this end, a couple of successful and relevant

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
0	1	0	1	0	0	1	1	1	0

Fig. 10 An encoded wolf after exploration changes by fourth selection

state-of-the-art approaches from each category are selected. The selected approaches are from the literature published recently in authentic journals to compete in comparison with the current presented hybrid DGWA (HyDGWA) approach. The selected literatures are heuristic filter-based (HF) [15], meta-heuristic-GWA (MGWA) [14], meta-heuristic-CSA (MCSA) [10], and hybrid meta-heuristic-NSGAII (HM-NSGAII) [9]. A little change has been made to selected approaches to customize them for competition in the same condition for reaching fair results. Two famous and prevalently being used datasets, i.e., NSL-KDD [18, 19, 43], and UNSW-NB15 [3, 8, 20, 44] in this context are also selected as real case benchmarks. As mentioned earlier, the used datasets have many features and samples. The main goal of the current paper is to generate a classifier model that detects the coming traffic whether is normal or not with a high rate of accuracy. Five comparative algorithms are executed on two famous datasets in the IDS context. Totally, 10 scenarios are defined. Except for a heuristic HF, other meta-heuristic approaches, i.e., proposed HyDGWA, MGWA, MCSA, and HM-NSGAII are run 30 times independently. Then, the average results of them are reported in terms of evaluation metrics in Tables 7 and 8. All of the scenarios have been executed in the same computing system, namely, a Laptop computer including a dual-core Intel Core i3380M with 2.53 GHz clock rate, four logical processors, 8 GB RAM, and Windows 10 platform running the Python programming language. To evaluate the performance of the work in the ML domain, a couple of prominent evaluation metrics are used: *accuracy*, *recall* (*or sensitivity*), *precision*, and *F-score* are incorporated which are calculated by Eq. (17) through Eq. (20), respectively [6, 16]. Another important parameter, *Specificity* (SP), which seldom is used is very important giving new insight [45]. This metric is obtained by Eq. (21). The *Specificity* (*SP*) is the true-negative rate (*TNR*). It calculates the number of correct negative predictions divided by the total number of exact negatives.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{17}$$

$$\text{percision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{18}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{19}$$

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
1	0	0	1	1	0	1	0	0	1

Fig. 11 An encoded Wolf after exploration changes by second selection

$$\text{F} - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{21}$$

All of them use TP, TN, FP, and FN parameters. The parameters are the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) that a classifier gives, respectively. For instance, the metric TN returns the number of negative labels where the classifier detects correctly the input cases are negative. In this paper, the user-defined parameter K is experimentally set to 20. In executing the proposed HyDGWA, for both datasets, three filter-based feature raking approaches are incorporated. The effectiveness of HyDGWA is tested in the two following parts.

Verifying the performance of HyDGWA by using the NSL-KDD dataset

In this part, the focus is on the NSL-KDD dataset. In the next episode, the same treatment for the UNSW-NB15 dataset is performed. For the NSL-KDD training dataset, after running Chi-Square, MAD, and PCC rankers; each of them assigns a priority (low/high value) to each feature. So, the alternative matrix $A_{41\times3}$ is created by Eq. (22). The rows are used for features whereas the columns are used for filter-based rankers.

$$A_{41\times 3} = \begin{array}{c} F./R. \\ F1 \\ F2 \\ F3 \\ F4 \\ F5 \\ F6 \\ F7 \\ F8 \\ F9 \\ F10 \\ F11 \\ F12 \\ F13 \\ F14 \\ F15 \\ F16 \\ F17 \\ F18 \\ F19 \\ F20 \\ F21 \\ F22 \\ F23 \\ F24 \\ F25 \\ F26 \\ F27 \\ F28 \\ F29 \\ F30 \\ F31 \\ F32 \\ F33 \\ F34 \\ F35 \\ F36 \\ F37 \\ F38 \\ F39 \\ F40 \\ F41 \end{array} \begin{bmatrix} ChiS & MAD & PCC \\ 0.57 & 0.33 & -0.79 \\ 0.62 & 0.34 & -0.76 \\ 0.87 & 0.33 & -0.73 \\ 0.76 & 0.36 & -0.80 \\ 0.95 & 228.71 & 0.011 \\ 0.94 & 189.56 & 0.007 \\ 0.55 & 0.63 & 0.003 \\ 0.54 & 0.63 & 0.037 \\ 0.54 & 0.63 & 0.026 \\ 0.57 & 0.77 & 0.060 \\ 0.53 & 0.63 & 0.147 \\ 0.52 & 0.97 & -0.508 \\ 0.57 & 0.87 & -0.008 \\ 0.50 & 0.63 & 0.027 \\ 0.50 & 0.63 & -0.018 \\ 0.48 & 0.90 & -0.009 \\ 0.45 & 0.64 & -0.008 \\ 0.45 & 0.63 & 0.029 \\ 0.44 & 0.63 & -0.023 \\ 0.42 & 0.63 & -0.600 \\ 0.42 & 0.63 & 0.029 \\ 0.57 & 0.63 & 0.126 \\ 0.42 & 82.89 & 0.296 \\ 0.63 & 27.81 & -0.064 \\ 0.57 & 0.38 & 0.289 \\ 0.41 & 0.38 & 0.245 \\ 0.68 & 0.53 & 0.296 \\ 0.67 & 0.53 & 0.295 \\ 0.70 & 0.97 & -0.418 \\ 0.41 & 0.61 & 0.285 \\ 0.70 & 0.66 & 0.006 \\ 0.40 & 183.32 & 0.183 \\ 0.80 & 118.92 & -0.552 \\ 0.80 & 0.95 & -0.460 \\ 0.81 & 0.59 & 0.355 \\ 0.74 & 0.62 & 0.259 \\ 0.41 & 0.63 & 0.198 \\ 0.59 & 0.39 & 0.289 \\ 0.39 & 0.38 & 0.291 \\ 0.77 & 0.58 & 0.280 \\ 0.72 & 0.54 & 0.297 \end{bmatrix} \quad (22)$$

Table 7 Performance comparison of comparative algorithms in terms of defined metrics using the NSL-KDD test dataset

Applied method [Ref.]	Accuracy	Recall	Precision	F-score	Specificity	Selected features
HF [15]	%91.10	%98.68	%96.79	%97.73	87.32%	['F0', 'F4', 'F5', 'F9', 'F25', 'F29', 'F31', 'F33', 'F34', 'F37', 'F38', 'F39']
MGWA [14]	%94.38	%99.65	%99.65	%99.56	86.22%	['F0', 'F5', 'F23', 'F29', 'F31', 'F32', 'F33', 'F35', 'F36', 'F37', 'F38', 'F39']
MCSA [10]	%93.42	%99.00	%97.49	%97.49	84.89%	['F11', 'F22', 'F24', 'F25', 'F28', 'F29', 'F35', 'F36', 'F38']
HM-NSGAII [9]	%93.04	%98.85	%97.35	%97.35	86.50%	['F6', 'F9',F23', 'F25', 'F28', 'F31', 'F33', 'F35', 'F36', 'F37', 'F38', 'F39']
Proposed HyDGWA	%99.11	%99.90	%99.88	%99.89	88.67%	['F0', 'F5', 'F9', 'F11', 'F25', 'F29', 'F31', 'F33', 'F34', 'F37']

Table 8 Performance comparison of comparative algorithms in terms of defined metrics using the UNSW_NB15 test dataset

Applied method [Ref.]	Accuracy	Recall	Precision	F-score	Specificity	Selected features
HF [15]	%84.28	%98.90	%98.45	%99.80	88.17%	'F1', 'F7', 'F8', 'F10', 'F13', 'F17', 'F18', 'F19', 'F21', 'F22', 'F24', 'F25', 'F27', 'F32']
MGWA [14]	%92.58	%99.95	%98.66	%99.97	90.69%	['F1', 'F6', 'F7','F13', 'F17', 'F19, ''F21', 'F27', 'F32', 'F39', 'F42']
MCSA [10]	%91.54	%98.27	%99.50	%99.13	90.94%	['F1', 'F7', 'F8', 'F10', 'F11', 'F12', 'F13', 'F16', 'F18', 'F21', 'F24', 'F25', 'F26', 'F32']
HM-NSGAII [9]	%91.49	%99.96	%98.80	%99.13	90.35%	['F0', 'F1', 'F9', 'F11', 'F12', 'F17', 'F19', 'F22', 'F24', 'F25', 'F32']
Proposed HyDGWA	%93.92	%99.98	%99.98	%99.78	91.33%	['F0', 'F7', 'F8', 'F11', 'F12', 'F13', 'F17', 'F21', 'F39', 'F42']

Since the importance of all rankers is taken the same, the weight vector is assigned to an orthogonal matrix $W_{3\times3}=\begin{bmatrix} 0.333 & 0 & 0 \\ 0 & 0.333 & 0 \\ 0 & 0 & 0.333 \end{bmatrix}$ each of them with 0.333. Then, the weighted alternative matrix ($WA_{41\times3}$) is created by Eq. (23); it should be normalized to avoid biased values.

$$WA_{41\times3}=\begin{bmatrix} Chi. & MAD & PCC \\ a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{43,1} & a_{43,2} & a_{43,3} \end{bmatrix}\times\begin{bmatrix} w_1 & w_2 & w_3 \\ 0.333 & 0 & 0 \\ 0 & 0.333 & 0 \\ 0 & 0 & 0.333 \end{bmatrix}=\begin{bmatrix} Chi. & MAD & PCC \\ wa_{1,1} & wa_{1,2} & wa_{1,3} \\ wa_{2,1} & wa_{2,2} & wa_{2,3} \\ wa_{43,1} & wa_{43,2} & wa_{43,3} \end{bmatrix} \tag{23}$$

$$WA_{41\times 3} = \begin{array}{c} F./R. \\ F1 \\ F2 \\ F3 \\ F4 \\ F5 \\ F6 \\ F7 \\ F8 \\ F9 \\ F10 \\ F11 \\ F12 \\ F13 \\ F14 \\ F15 \\ F16 \\ F17 \\ F18 \\ F19 \\ F20 \\ F21 \\ F22 \\ F23 \\ F24 \\ F25 \\ F26 \\ F27 \\ F28 \\ F29 \\ F30 \\ F31 \\ F32 \\ F33 \\ F34 \\ F35 \\ F36 \\ F37 \\ F38 \\ F39 \\ F40 \\ F41 \end{array} \begin{bmatrix} ChiS & MAD & PCC \\ 0.19 & 0.11 & -0.26333 \\ 0.206667 & 0.113333 & -0.25333 \\ 0.29 & 0.11 & -0.24333 \\ 0.25333 & 0.12 & -0.26667 \\ 0.316667 & 76.23667 & 0.03667 \\ 0.31333 & 63.18667 & 0.002333 \\ 0.18333 & 0.21 & 0.001 \\ 0.18 & 0.21 & 0.012333 \\ 0.18 & 0.21 & 0.008667 \\ 0.19 & 0.256667 & 0.02 \\ 0.176667 & 0.21 & 0.049 \\ 0.173333 & 0.323333 & -0.16933 \\ 0.19 & 0.29 & -0.00267 \\ 0.166667 & 0.21 & 0.009 \\ 0.166667 & 0.21 & -0.006 \\ 0.16 & 0.3 & -0.003 \\ 0.15 & 0.213333 & -0.00267 \\ 0.15 & 0.21 & 0.009667 \\ 0.146667 & 0.21 & -0.00767 \\ 0.14 & 0.21 & -0.2 \\ 0.14 & 0.21 & 0.009667 \\ 0.19 & 0.21 & 0.042 \\ 0.14 & 27.63 & 0.098667 \\ 0.21 & 9.27 & -0.02133 \\ 0.19 & 0.126667 & 0.096333 \\ 0.136667 & 0.126667 & 0.081667 \\ 0.226667 & 0.176667 & 0.098667 \\ 0.223333 & 0.176667 & 0.098333 \\ 0.233333 & 0.323333 & -0.13933 \\ 0.136667 & 0.203333 & 0.095 \\ 0.233333 & 0.22 & 0.002 \\ 0.133333 & 61.10667 & 0.061 \\ 0.266667 & 39.64 & -0.184 \\ 0.266667 & 0.316667 & -0.15333 \\ 0.27 & 0.196667 & 0.118333 \\ 0.246667 & 0.206667 & 0.086333 \\ 0.136667 & 0.21 & 0.066 \\ 0.196667 & 0.13 & 0.096333 \\ 0.13 & 0.126667 & 0.097 \\ 0.256667 & 0.19333 & 0.093333 \\ 0.24 & 0.18 & 0.099 \end{bmatrix} \quad (24)$$

The normalized decision relative matrix $DR_{41\times3}$ is calculated in Eq. (24) using Eq. (13) on each item.

$$DR_{41\times3} = \begin{array}{c|ccc} F./R. & ChiS & MAD & PCC \\ F1 & 0.144909 & 0.000871 & -0.35975 \\ F2 & 0.157621 & 0.000897 & -0.34609 \\ F3 & 0.221177 & 0.000871 & -0.33243 \\ F4 & 0.193212 & 0.00095 & -0.36431 \\ F5 & 0.241515 & 0.603443 & 0.005009 \\ F6 & 0.238973 & 0.500147 & 0.003188 \\ F7 & 0.139825 & 0.001662 & 0.001366 \\ F8 & 0.137282 & 0.001662 & 0.016849 \\ F9 & 0.137282 & 0.001662 & 0.01184 \\ F10 & 0.144909 & 0.002032 & 0.027323 \\ F11 & 0.13474 & 0.001662 & 0.066941 \\ F12 & 0.132198 & 0.002559 & -0.23133 \\ F13 & 0.144909 & 0.002295 & -0.00364 \\ F14 & 0.127113 & 0.001662 & 0.012295 \\ F15 & 0.127113 & 0.001662 & -0.0082 \\ F16 & 0.122029 & 0.002375 & -0.0041 \\ F17 & 0.114402 & 0.001689 & -0.00364 \\ F18 & 0.114402 & 0.001662 & 0.013206 \\ F19 & 0.11186 & 0.001662 & -0.01047 \\ F20 & 0.106775 & 0.001662 & -0.27323 \\ F21 & 0.106775 & 0.001662 & 0.013206 \\ F22 & 0.144909 & 0.001662 & 0.057378 \\ F23 & 0.106775 & 0.218702 & 0.134793 \\ F24 & 0.160163 & 0.073376 & -0.02914 \\ F25 & 0.144909 & 0.001003 & 0.131605 \\ F26 & 0.104233 & 0.001003 & 0.111568 \\ F27 & 0.172874 & 0.001398 & 0.134793 \\ F28 & 0.170332 & 0.001398 & 0.134338 \\ F29 & 0.177959 & 0.002559 & -0.19035 \\ F30 & 0.104233 & 0.001609 & 0.129784 \\ F31 & 0.177959 & 0.001741 & 0.002732 \\ F32 & 0.101691 & 0.483683 & 0.083335 \\ F33 & 0.203381 & 0.313766 & -0.25137 \\ F34 & 0.203381 & 0.002507 & -0.20948 \\ F35 & 0.205924 & 0.001557 & 0.16166 \\ F36 & 0.188128 & 0.001636 & 0.117944 \\ F37 & 0.104233 & 0.001662 & 0.090166 \\ F38 & 0.149994 & 0.001029 & 0.131605 \\ F39 & 0.099148 & 0.001003 & 0.132516 \\ F40 & 0.195755 & 0.00153 & 0.127507 \\ F41 & 0.183043 & 0.001425 & 0.135248 \end{array} \quad (25)$$

Two *three*-dimensional positive ideal solutions: $A^+ = [0.2415, 0.6034, 0.1617]$ and negative ideal solution $A^- = [0.0991, 0.0009, -0.3643]$ are calculated from Eq. (24). By using Eq. (13) and Eq. (14), $S^+_{1\times41}$=[0.803, 0.792, 0.780, 0.801, 0.157, 0.189, 0.631, 0.628, 0.629, 0.624, 0.618, 0.726, 0.631, 0.631, 0.636, 0.635, 0.637, 0.633, 0.639, 0.755, 0.634, 0.618, 0.409, 0.569, 0.611, 0.620, 0.607, 0.607, 0.699, 0.618, 0.626, 0.200, 0.506, 0.707, 0.603, 0.606, 0.621, 0.610, 0.620, 0.605, 0.605] and $S^-_{1\times41}$=[0.046, 0.061, 0.126, 0.094, 0.721, 0.636, 0.368, 0.383, 0.378, 0.394, 0.433, 0.137, 0.364, 0.378, 0.357, 0.361, 0.361, 0.378, 0.354, 0.091, 0.378, 0.424, 0.545, 0.348, 0.498, 0.476, 0.505, 0.504, 0.191, 0.494, 0.375, 0.658, 0.349, 0.187, 0.537, 0.490, 0.454, 0.499, 0.497, 0.501, 0.507] are calculated. By using Eq. (16), the closeness matric $C_{1\times41}$=[0.054, 0.072, 0.139, 0.105, 0.821, 0.771, 0.368, 0.379, 0.375, 0.387, 0.412, 0.159, 0.366, 0.375, 0.360, 0.362, 0.362, 0.374, 0.356, 0.108, 0.373, 0.407, 0.571, 0.380, 0.449, 0.434, 0.454, 0.454, 0.215, 0.444, 0.375, 0.767, 0.408, 0.209, 0.471, 0.447, 0.422, 0.450, 0.445, 0.453, 0.456] is calculated by Eq. (16). If the values in $C_{1\times41}$ are sorted in decreasing order; finally, the ranking list of features are in from the most important to the less: [4, 5, 31, 22, 34, 40, 26, 27, 39, 37, 24, 35, 38, 29, 25, 36, 10, 32, 21, 9, 23, 7, 8, 30, 13, 17, 20, 6, 12, 15, 16, 14, 18, 28, 33, 11, 2, 19, 3, 1, 0]. Therefore, the top-K ($K=20$) list of features is [4, 5, 9, 10, 27, 28, 33, 34, 35, 36, 46, 13, 14, 43, 44, 29, 30, 31, 38, 22].

After sorting in descending order, K (=20) out of N (=41) features are selected as a top-K subset that is [$Chr_1 = F_4$,$Chr_2 = F_5$,$Chr_3 = F_{31}$,$Chr_4 = F_{22}$,$Chr_5 = F_{34}$,…,$Chr_{20} = F_9$]. The proposed discrete DGWA is called to potentially condense this subset gained from the TOPSIS approach and also to improve the accuracy metric. In other words, when the swarm of wolves is encoded in a K-length chromosome-like representation, the binary value determines which feature should be considered for a final optimal set. In fact, DGWA works on features [F_4,F_5,F_{31} ,F_{22}, F_{34}…,F_9] and ignores others because of their low importance on the whole. This procedure is iterated until the significant changes are not achieved based on fitness function (accuracy). For instance, the encoded wolf WL_1=[1,1,0,0,1 …] means that features:F_4,F_5,F_{34}, … are considered in feature set and F_{31} andF_{22}, etc. are omitted. Finally, the classifier is made based on this optimal feature set.

Figure 12 shows all confusion matrices associated with comparative algorithms. Note that label "0" is dedicated for normal connections whereas the labels 1, 2, 3, and 4 are, respectively, considered for DoS, Probing, U2R, and R2L attacks.

Figure 13 illustrates the performance comparisons schematically in terms of evaluation metrics in the ML context using the NSL_KDD dataset.

Figure 13 provides schematic information about the significant superiority of the proposed algorithm. Since the mentioned discrete problem is NP-Hard, the majority of literature ignores possible permutations. The proposed discrete GWA efficiently permutes search space by calling its local and global search operators. In the course of optimization, it selects a better feature set which improves accuracy that was considered as an objective function. Also, Table 7 provides the performance the comparison among all of comparative algorithms in terms of

accuracy, *recall*, *precision*, and *F-score* using the NSL-KDD testing dataset. In addition, the set of selected features by each algorithm is shown in Table 7

A closer look at the information in Table 7 informs the proposed HyDGWA outperforms other comparative algorithms in terms of all evaluation metrics. After HyDGWA that is the best, MGWA, MCSA, HM-NSGAII, and HF are in second to fifth place in terms of accuracy. Except for some changes, the list of better to worst is the same in terms of other assessment metrics. To measure the amount of performance improvements against other counterparts, the relative percentage deviation (RPD) metric is derived from literature [47]. By incorporation of RPD metric, the proposed HyDGWA outperforms about 9.07%, 14.41%, 2.28%, 5.50%, and 2.78% improvement against other counterparts, respectively, in terms of *accuracy*, *recall*, *precision*, *F-score*, and *specificity* metrics in using NSL-KDD testing dataset. One important thing about the values in the column of "*specificity*" in Table 7 is that its performance is lower than in comparison with other metrics. The proposed HyDGWA marginally outperforms against other state of the art in terms of *specificity*. This revolves around the fact that the fitness function has a direct relationship with true positive and neglects true-positive rate. For this reason, the metrics that have a direct relationship with true positives have better results.

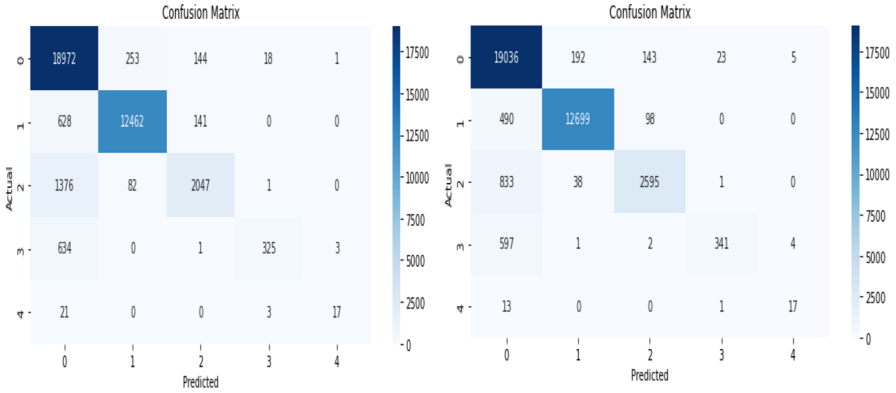
Verifying the performance of HyDGWA by using the UNSW_NB15 dataset

In this part, the focus is on the UNSW-NB15 dataset. After running Chi-square, MAD, and PCC rankers; each of them assigns a priority (low/high value) to each feature. So, the alternative matrix $A_{44\times3}$ is created by Eq. (25) since it has 44 features.

$$A_{44\times 3} = \begin{bmatrix} F./R. & ChiS & MAD & PCC \\ F1 & 1.000 & 87670.32 & 0.727173 \\ F2 & 0.915669 & 1.641882 & 0.036175 \\ F3 & 0.18913 & -900000 & -900000 \\ F4 & 0.073846 & -900000 & -900000 \\ F5 & 0.312191 & -900000 & -900000 \\ F6 & 0.350369 & 19.61804 & -0.05218 \\ F7 & 0.513238 & 19.1744 & -0.11859 \\ F8 & 0.790793 & 8844.163 & 0.018576 \\ F9 & 0.0670061 & 14929.12 & -0.07687 \\ F10 & 0.91367 & 95405.52 & 0.337979 \\ F11 & 0.677007 & 178.8698 & 0.692741 \\ F12 & 0.629227 & 79.81437 & 0.095049 \\ F13 & 0.955067 & 73454033 & 0.18287 \\ F14 & 0.838032 & 671205.8 & -0.39374 \\ F15 & 0.239168 & 5.16838 & -0.00064 \\ F16 & 0.289178 & 7.165865 & -0.09468 \\ F17 & 0.757949 & 986.1399 & -0.17611 \\ F18 & 0.822256 & 88.42856 & -0.02289 \\ F19 & 0.690729 & 4976.423 & -0.00707 \\ F20 & 0.662416 & 604.5671 & -0.06087 \\ F21 & 0.111243 & 116.4723 & -0.33363 \\ F22 & 0.620431 & 9.69E+8 & -0.25501 \\ F23 & 0.620327 & 9.69E+8 & -0.25034 \\ F24 & 0.102021 & 115.2286 & -0.31963 \\ F25 & 0.600185 & 0.659643 & 0.081584 \\ F26 & 0.588495 & 0.67067 & 0.058299 \\ F27 & 0.580897 & 0.669594 & 0.097364 \\ F28 & 0.58988 & 136.0711 & -0.0108 \\ F29 & 0.579287 & 124.3782 & -0.34181 \\ F30 & 0.000926 & 0.6422 & 0.010801 \\ F31 & 0.061582 & 2144.897 & -0.02136 \\ F32 & 0.146036 & 8.625815 & 0.229044 \\ F33 & 0.637638 & 0.63364 & 0.577704 \\ F34 & 0.190315 & 5.513314 & 0.229887 \\ F35 & 0.132686 & 4.702916 & 0.305579 \\ F36 & 0.196242 & 3.525633 & 0.357213 \\ F37 & 0.153367 & 8.049258 & 0.303855 \\ F38 & 0.000 & 0.677115 & -0.01105 \\ F39 & 0.000 & 0.677115 & -0.01105 \\ F40 & 0.000547 & 0.669284 & 0.0158 \\ F41 & 0.153635 & 6.275167 & 0.238225 \\ F42 & 0.171382 & 8.420136 & 0.228046 \\ F43 & 0.033865 & 0.696374 & -0.18468 \\ F44 & 1.000 & -900000 & -900000 \end{bmatrix} \quad (26)$$

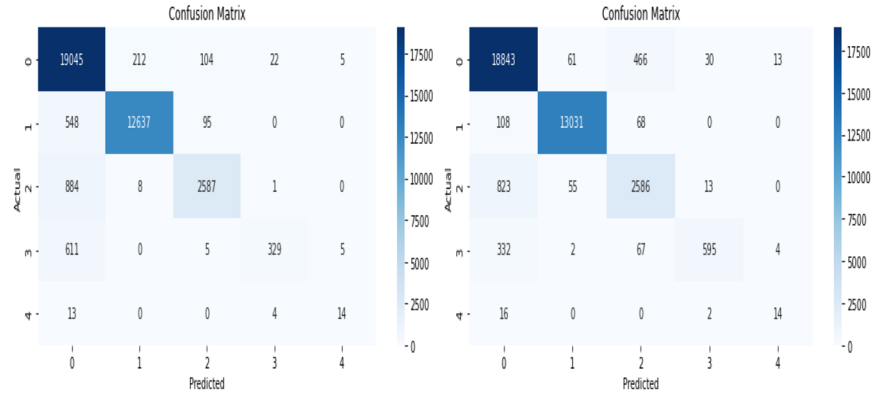
Since the importance of all rankers is taken the same, the weight vector is assigned to an orthogonal matrix $W_{3\times3}=\begin{bmatrix} 0.333 & 0 & 0 \\ 0 & 0.333 & 0 \\ 0 & 0 & 0.333 \end{bmatrix}$ each of them with 0.333. Then, the weighted alternative matrix ($WA_{44\times3}$) is created by Eq. (26); it should be normalized to avoid biased values.

$$WA_{41\times3}=\begin{bmatrix} Chi. & MAD & PCC \\ a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{44,1} & a_{44,2} & a_{44,3} \end{bmatrix}\times\begin{bmatrix} w_1 & w_2 & w_3 \\ 0.333 & 0 & 0 \\ 0 & 0.333 & 0 \\ 0 & 0 & 0.333 \end{bmatrix}=\begin{bmatrix} Chi. & MAD & PCC \\ wa_{1,1} & wa_{1,2} & wa_{1,3} \\ wa_{2,1} & wa_{2,2} & wa_{2,3} \\ wa_{44,1} & wa_{44,2} & wa_{44,3} \end{bmatrix} \tag{27}$$



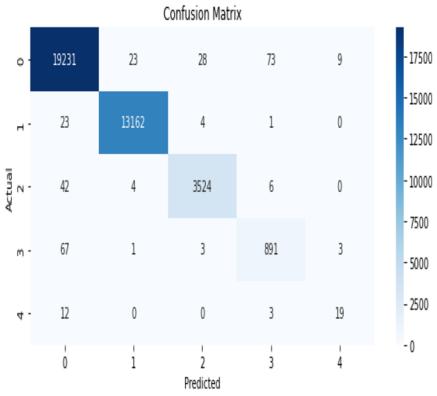
a. HF [13] gives 91% accuracy

b. MCSA [10] gives 93% accuracy



c. NSGA-II [9] gives 93% accuracy

d. MGWA [32] gives 94% accuracy



e. HyDGWA gives 99% accuracy

Fig. 12 Illustration of confusion matrices associated with all comparative algorithms in NSL_KDD dataset

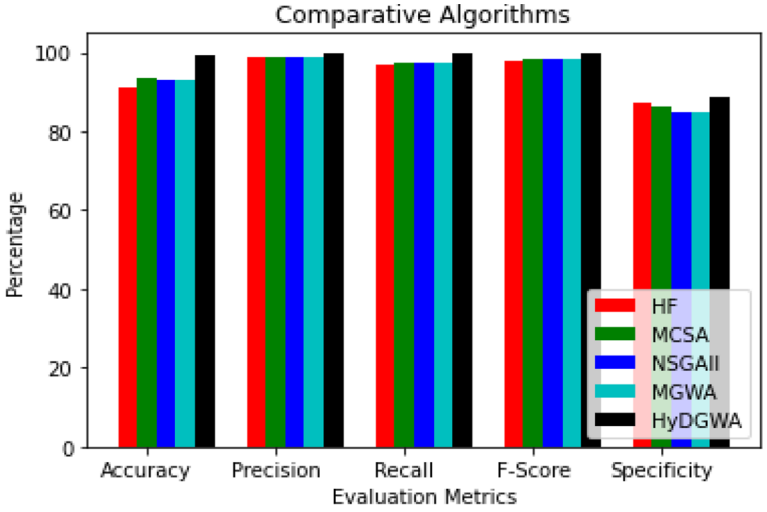


Fig. 13 Performance comparison of comparative algorithms in terms of assessment parameters using the NSL_KDD

$$WA_{44\times3} = \begin{array}{c|ccc} F./R. & ChiS & MAD & PCC \\ F1 & 0.333333 & 29223.44 & 0.242391 \\ F2 & 0.305223 & 0.547294 & 0.012058 \\ F3 & 0.63043 & -300000 & -300000 \\ F4 & 0.024615 & -300000 & -300000 \\ F5 & 0.104064 & -300000 & -300000 \\ F6 & 0.11679 & 6.539347 & -0.01739 \\ F7 & 0.171079 & 6.391466 & -0.03953 \\ F8 & 0.263598 & 2948.054 & 0.006192 \\ F9 & 0.223354 & 4976.374 & -0.02562 \\ F10 & 0.304557 & 31801.84 & 0.11266 \\ F11 & 0.225669 & 59.62328 & 0.230914 \\ F12 & 0.209742 & 26.60479 & 0.031683 \\ F13 & 0.318356 & 24484678 & 0.060957 \\ F14 & 0.279344 & 223735.3 & -0.13125 \\ F15 & 0.079723 & 1.722793 & -0.00021 \\ F16 & 0.096393 & 2.388622 & -0.03156 \\ F17 & 0.25265 & 328.7133 & -0.0587 \\ F18 & 0.274085 & 29.47619 & -0.00763 \\ F19 & 0.230243 & 1658.808 & -0.00236 \\ F20 & 0.220805 & 201.5224 & -0.02029 \\ F21 & 0.037081 & 38.3241 & -0.11121 \\ F22 & 0.20681 & 3.23E+08 & -0.085 \\ F23 & 0206776 & 38.40953 & -0.08345 \\ F24 & 0.034007 & 0.219881 & -0.10654 \\ F25 & 0.200062 & 0.223557 & 0.027195 \\ F26 & 0.196165 & 0.223198 & 0.019433 \\ F27 & 0.193632 & 45.35705 & 0.032455 \\ F28 & 0.196627 & 41.4594 & -0.0036 \\ F29 & 0.193096 & 0.214067 & -0.11394 \\ F30 & 0.000309 & 714.9658 & 0.0036 \\ F31 & 0.020527 & 2.875272 & -0.00712 \\ F32 & 0.048679 & 0.211213 & 0.076348 \\ F33 & 0.212546 & 1.837771 & 0.192568 \\ F34 & 0.063438 & 1.567639 & 0.076629 \\ F35 & 0.044229 & 1.175211 & 0.10186 \\ F36 & 0.065414 & 2.683086 & 0.119071 \\ F37 & 0.051122 & 0.225705 & 0.101285 \\ F38 & 0.00 & 0.225705 & -0.00368 \\ F39 & 0.00 & 0.223095 & -0.00368 \\ F40 & 0.000182 & 2.091722 & 0.005267 \\ F41 & 0.051212 & 2.806712 & 0.079408 \\ F42 & 0.057127 & 0.232125 & 0.076015 \\ F43 & 0.011288 & -300000 & -0.06156 \\ F44 & 0.333333 & & -300000 \end{array} \quad (28)$$

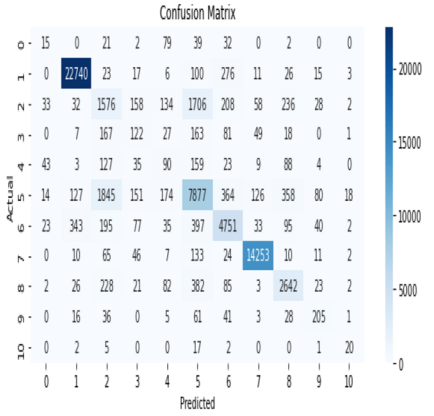
The normalized decision relative matrix $DR_{44\times3}$ is calculated in Eq. (27) using Eq. (13) on each item.

$$DR_{44\times 3} = \begin{bmatrix} F./R. & ChiS & MAD & PCC \\ F1 & 0.275552 & 6.39E-05 & 4.04E-07 \\ F2 & 0.252315 & 1.20E-09 & 2.01E-08 \\ F3 & 0.052115 & -0.00066 & -0.5 \\ F4 & 0.020348 & -0.00066 & -0.5 \\ F5 & 0.086025 & -0.00066 & -0.5 \\ F6 & 0.096545 & 1.43E-08 & -2.90E-08 \\ F7 & 0.141424 & 1.40E-08 & -6.59E-08 \\ F8 & 0.217905 & 6.44E-06 & 1.03E-08 \\ F9 & 0.184637 & 1.09E-05 & -4.27E-08 \\ F10 & 0.251764 & 6.95E-05 & 1.88E-07 \\ F11 & 0.186551 & 1.30E-07 & 3.85E-07 \\ F12 & 0.173385 & 5.82E-08 & 5.28E-08 \\ F13 & 0.263171 & 0.053521 & 1.02E-07 \\ F14 & 0.230922 & 0.000489 & -2.19E-07 \\ F15 & 0.065903 & 3.77E-09 & -3.56E-10 \\ F16 & 0.079684 & 5.22E-09 & -5.26E-08 \\ F17 & 0.208855 & 7.19E-07 & -9.78E-08 \\ F18 & 0.226574 & 6.44E-08 & -1.27E-08 \\ F19 & 0.190332 & 3.63E-06 & -3.93E-09 \\ F20 & 0.18253 & 4.41E-07 & -3.38E-08 \\ F21 & 0.030653 & 8.49E-08 & -1.85E-07 \\ F22 & 0.170961 & 0.706229 & -1.42E-07 \\ F23 & 0.170932 & 0.705957 & -1.39E-07 \\ F24 & 0.028112 & 8.40E-08 & -1.78E-07 \\ F25 & 0.165382 & 4.81E-10 & 4.53E-08 \\ F26 & 0.162543 & 4.89E-10 & 3.24E-08 \\ F27 & 0.160067 & 4.88E-10 & 5.41E-08 \\ F28 & 0.162543 & 9.91E-08 & -6.00E-09 \\ F29 & 0.159624 & 9.06E-08 & -1.90E-07 \\ F30 & 0.000255 & 4.68E-10 & 6.00E-09 \\ F31 & 0.016969 & 1.56E-06 & -1.19E-08 \\ F32 & 0.040241 & 6.29E-09 & 1.27E-07 \\ F33 & 0.175703 & 4.62E-10 & 3.21E-07 \\ F34 & 0.052442 & 4.02E-09 & 1.28E-07 \\ F35 & 0.036562 & 3.43E-09 & 1.70E-07 \\ F36 & 0.054075 & 2.57E-09 & 1.98E-07 \\ F37 & 0.042261 & 5.86E-09 & 1.69E-07 \\ F38 & 0.00 & 4.93E-10 & -6.14E-09 \\ F39 & 0.00 & 4.93E-10 & -6.14E-09 \\ F40 & 0.000151 & 4.88E-10 & 8.78E-09 \\ F41 & 0.042334 & 4.57E-09 & 1.32E-07 \\ F42 & 0.047225 & 6.14E-09 & 1.27E-07 \\ F43 & 0.009332 & 5.07E-10 & -1.03E-07 \\ F44 & 0.275552 & -0.00066 & -0.5 \end{bmatrix} \quad (29)$$

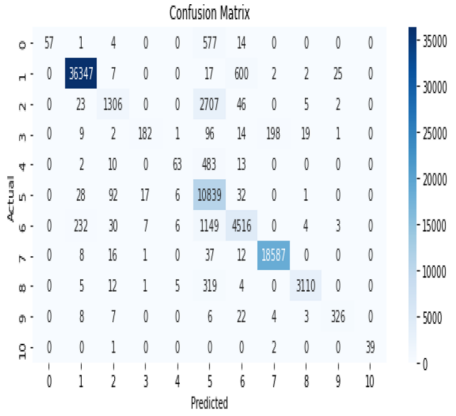
Two *three*-dimensional positive ideal solutions: $A^+ = [0.275552163, 0.706228623, 4.03984973e-07]$ and negative ideal solution $A^- = [0.0, -0.00065577, -0.5]$ are calculated from Eq. (24). By using Eq. (13) and Eq. (14), $S^+_{1\times44}$=[0.70616474, 0.70661082, 0.89420918, 0.90267102, 0.8863445, 0.72856193, 0.71885276, 0.7085711, 0.71204571, 0.70655968, 0.71181453, 0.71358037, 0.65282495, 0.70714935, 0.73668959, 0.73288696, 0.70937044, 0.70792485, 0.71134819, 0.71232812, 0.74748527, 0.10459096, 0.10462019, 0.7483217, 0.71477007, 0.71527364, 0.71560854, 0.71521317, 0.71568018, 0.75798894, 0.75207837, 0.74439935, 0.71325228, 0.74063295, 0.7455704, 0.74014255, 0.74376325, 0.7580817, 0.7580817, 0.75802696, 0.74374013, 0.74222116, 0.75473985, 0.86584407] and $S^-_{1\times44}$=[0.57090272, 0.56005633, 0.05211511, 0.02034829, 0.08602481, 0.509236, 0.51961623, 0.54541991, 0.53300204, 0.55980858, 0.53366845, 0.52920959, 0.56762147, 0.55075031, 0.50432496, 0.50631006, 0.54186768, 0.54894115, 0.53500155, 0.5322759, 0.50093898, 0.88256056, 0.88233707, 0.50078992, 0.52664201, 0.52563933, 0.52499721, 0.5257571, 0.52486189, 0.5000005, 0.50028829, 0.50161725, 0.52997372, 0.50274316, 0.50133559, 0.50291623, 0.50178338, 0.50000042, 0.50000042, 0.50000046, 0.50178956, 0.50222578, 0.5000874, 0.27555216] are calculated. By using Eq. (16), the closeness matric $C_{1\times44}$=[0.44704194, 0.44214957, 0.05507109, 0.02204536, 0.08846928, 0.41140479, 0.41956338, 0.43494723, 0.42809767, 0.44205829, 0.42848314, 0.42582384, 0.46509332, 0.43783326, 0.40638118, 0.40857915, 0.4330652, 0.43675392, 0.42925475, 0.42766687, 0.40125701, 0.89404771, 0.89399725, 0.40091687, 0.4242282, 0.42359081, 0.42317812, 0.42366615, 0.42309076, 0.39746001, 0.39947429, 0.4025767, 0.42628912, 0.40433716, 0.40206366, 0.40457961, 0.40286198, 0.39743067, 0.39743067, 0.39744798, 0.40287242, 0.40357348, 0.39853087, 0.24141675] is calculated by Eq. (16). After sorting the array $C_{1\times44}$ in descending order, the list [21, 22, 12, 13, 9, 0, 8, 7, 18, 30, 10, 16, 32, 19, 35, 36, 34, 40, 31,41] is obtained that is K (=20) out of N (=44) features. The K-length chromosome that is [$Chr_1 = F_{21}$,$Chr_2 = F_{22}$,$Chr_3 = F_{12}$,$Chr_4 = F_{13}$,…,$Chr_{20} = F_{41}$] is delivered to discrete GWA (DGWA) to optimize it. After running all programs in the same conditions, Fig. 14 shows the confusion matrices associated with all comparative algorithms. In Fig. 14, the label "0" is dedicated to normal connections whereas the labels 1, 2, 3,…, 8, and 9 are, respectively, considered for "DoS," "Analysis," "Backdoor," "Exploits," "Fuzzers," "Generic," "Reconnaissance," "Shellcode," and "Worms" attacks.

Table 8 provides the performance comparison among all of the comparative algorithms in terms of *accuracy*, *recall*, *precision*, and *F-score* using the UNSW_NB15 testing dataset. Also, Table 8 gives the selected features by each comparative algorithm.

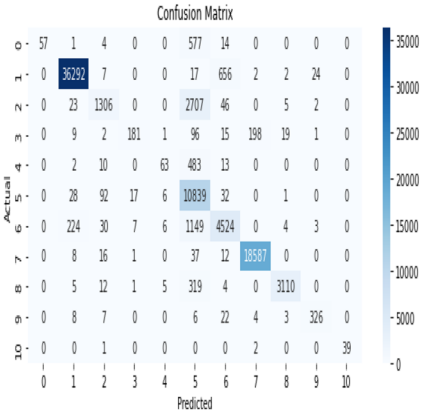
To measure the amount of performance improvements against other counterparts, the relative percentage deviation (RPD) metric is also used [47]. By incorporating of RPD metric, the proposed HyDGWA outperforms about 13.12%, 17.29%, 4.32%, 3.28%, and 1.28% improvement against other counterparts, respectively, in terms of *accuracy*, *recall*, *precision*, *F-score*, and *specificity* metrics in using the UNSW_NB15 testing dataset. Figure 15 illustrates the performance comparisons



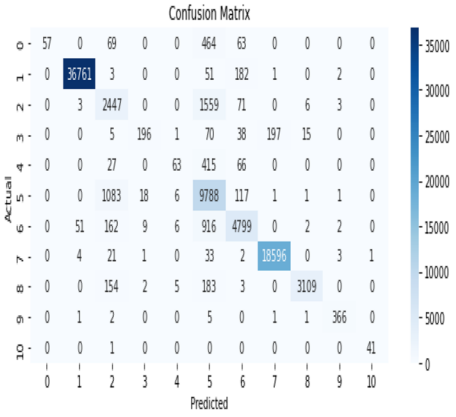
a. HF [13] gives 84.28% accuracy



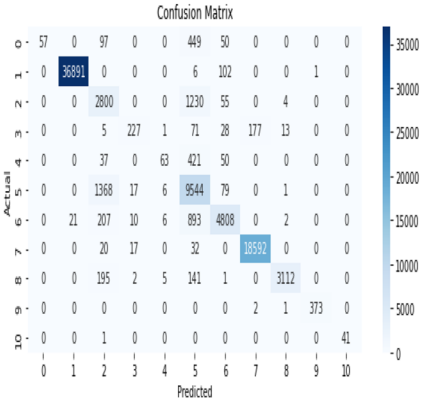
b. MCSA [10] gives 91.54% accuracy



c. NSGA-II [9] gives 91.49% accuracy



d. MGWA [32] gives 92.58% accuracy



e. HyDGWA gives 92.92% accuracy

Fig. 14 Illustration of confusion matrices associated with all comparative algorithms in the UNSW_NB15 dataset

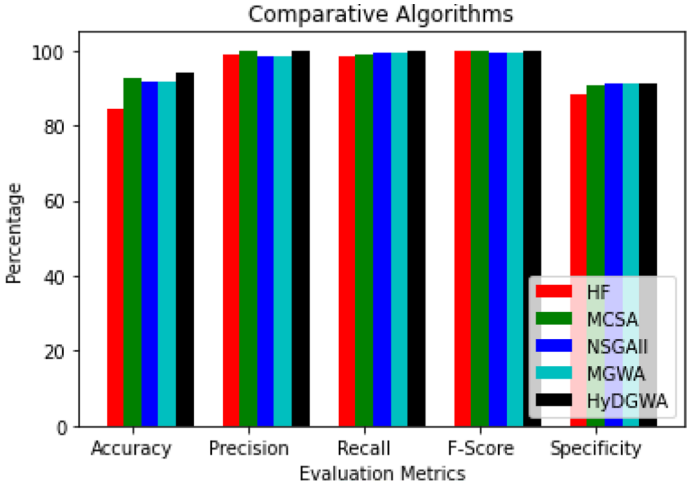


Fig. 15 Performance comparison of all algorithms in terms of assessment parameters using the UNSW_NB15

schematically in terms of evaluation metrics in the ML context. One important thing about the values in the column of "*specificity*" in Table 8 is that its performance is lower than in comparison with other metrics. The proposed HyDGWA marginally outperforms against other state of the art in terms of *specificity*. This revolves around the fact that the fitness function has a direct relationship with true positive and neglects true-positive rate. For this reason, the metrics that have a direct relationship with true positives have better results.

Figure 15 depicts the performance comparison of all comparative algorithms. The bar chart associated with the proposed HyDGWA algorithm has a better illustration in terms of all evaluation metrics. As mentioned earlier, the last phase of the proposed algorithm, i.e., permute discrete chromosomes efficiently. This significant superiority revolves around the fact that a good balance between exploration and exploitation is done to optimize the gained solutions. This tuning between local search and global search avoids the early convergence phenomenon; therefore, the quality of solutions is gradually improved.

7 Discussion, potentials, and limitations

For the sake of low overhead, heuristic-based feature selection approaches are usually recommended in the ML context in comparison with meta-heuristic-based approaches. Among the feature selection approaches, the wrapper and hybrid approaches return rather better solutions, but they impose more computation costs. One of the most important challenges is associated with high complexity in real-time applications such as IoT applications in distributed systems. To this end, filter-based approaches are recommended. Nevertheless, since the input datasets are naturally nonlinear and also multi-feature sets increase the computation complexity,

selecting the most efficient filter approach among the available filter-based feature selection approaches is a challenging task. Since there is clear diversity in datatype and a number of features in given datasets whether financial datasets, medicine datasets, or information technology datasets, the undiscovered complex and confusing dependencies and relationships are there that should be diagnosed. For this reason, the performance of filter heuristics should be experimentally investigated for a given dataset for the final decision which increases its challenges and limitations. To this end, for IDS datasets majority of available feature selection approaches were examined; then, the three most effective ones were selected. To have a consensus between the three, a strong decision-maker fuzzy TOPSIS was customized to return the top-*K* features. Since the problem is NP-Hard and this gained a set of top-*K* features is not the optimal solution for the IDS model, to potentially improve the outcome, a new discrete version of GWA was extended. The original GWA successfully makes an unbiased balance in local and global traversing in the search space that potentially returns promising solutions. All of the plus points of the original GWA help to design an original discrete version of GWA. In this regard, presenting a new exploration algorithm in Algorithm 4 that uses diverse fashions for global search makes efficient permutations of discrete search space. It leads to finding promising solutions in some promising areas that other approaches may not discover. In addition, utilizing the experience of leader wolves strengthens the local search inclination. This proposed lightweight and low-overhead HyDGWA algorithm returns the most efficient subset of features in making an efficient classifier that detects attacks in distributed systems efficiently. As another challenge and limitation, since the proposed IDS model takes available features into account, it may have not a good reaction against new types of unknown attacks.

8 Conclusion and future direction

This paper presented a hybrid discrete gray wolf optimization algorithm (HyDGWA) that is a four-phase filter selection algorithm to generate a classifier detection model for IDS networks in distributed systems. In the first phase, preprocessing is done to shed outliers and to normalize the given datasets. Then, in training process has two phases that are to call heuristic-based and meta-heuristic-based approaches. In the heuristic phase, a couple of filter-based rankers are called; then, the fuzzy TOPSIS returns the top_*K* consensus of the feature set. In the third phase, a discrete gray wolf optimization algorithm (DGWA) was presented to return the optimal IDS classifier model based on optimal feature selection among top_*K* features. Finally, in the fourth phase, the generated model is tested on a couple of relevant datasets such as NSL_KDD and UNSW_NB15 datasets. By incorporating of RPD metric, the proposed HyDGWA outperforms about 13.12%, 17.29%, 4.32%, 3.28%, and 1.28% improvement against other counterparts, respectively, in terms of *accuracy*, *recall*, *precision*, *F-score*, and *specificity* metrics in using UNSW_NB15 testing dataset. Also, the proposed HyDGWA outperforms about 9.07%, 14.41%, 2.28%, 5.50%, and 2.78% improvement against other counterparts, respectively, in terms of *accuracy*, *recall*, *precision*, *F–score*, and *specificity* metrics in using the NSL-KDD testing

dataset. Totally, in different 10 scenarios, the proposed HyDGWA model gives an average of 10.60%, 15.85%, 3.30%, 4.39%, and 2.03% improvement in testing datasets in terms of accuracy, precision, recall, *F–score*, and *specificity* against other comparative state of the art in the same conditions. For future work, we envisage introducing a new measure and an intricate heuristic filter-based approach to assign a weight to each feature for preparing a semi-conducted initial population of solutions of a strong meta-heuristic-based classifier.

Author Contribution Yashar Pourardebil Khah contributed to writing and programming; Mirsaeid Hosseini Shirvani contributed to algorithm design, supervisor, data analysis, and editing; and Homayun Motameni contributed to advisor and editing.

Data availability No datasets were generated or analyzed during the current study.

Declarations

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

Conflict of interest The authors declare no competing interests.

References

1. Ayad AG, Sakr NA, Hikal NA (2024) A hybrid approach for efficient feature selection in anomaly intrusion detection for IoT networks. J Supercomput. https://doi.org/10.1007/s11227-024-06409-x
2. Sanju P (2023) Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks. J Eng Res 11:356–361. https://doi.org/10.1016/j.jer.2023.100122
3. Moustafa N, Slay J, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 2015;1-6, https://doi.org/10.1109/MilCIS.2015.7348942.
4. Hosseini Shirvani M, Masdari M (2023) A survey study on trust-based security in Internet of Things: challenges and issues. Internet Things 21:100640. https://doi.org/10.1016/j.iot.2022.100640
5. Siddiqi MA, Pak W (2020) Optimizing filter-based feature selection method flow for intrusion detection system. Electronics 9(12):2114. https://doi.org/10.3390/electronics9122114
6. Han J, Kamber M, Pei J, Data Mining Concepts and Techniques Third Edition, AMSTERDAM _ BOSTON _ HEIDELBERG _ LONDON NEWYORK _ OXFORD _ PARIS _ SAN DIEGO SAN FRANCISCO _ SINGAPORE _ SYDNEY _ TOKYO Morgan Kaufmann is an imprint of Elsevier.
7. Halim Z, Yousaf MN, Waqas M, Sulaiman M, Abbas G, Hussain M, Ahmad I, Hanif M (2021) An effective genetic algorithm-based feature selection method for intrusion detection systems. Comput Secur 110:102448. https://doi.org/10.1016/j.cose.2021.102448
8. Maseno EM, Wang Z (2024) Hybrid wrapper feature selection method based on genetic algorithm and extreme learning machine for intrusion detection. J Big Data 11:24. https://doi.org/10.1186/s40537-024-00887-9
9. Dey AK, Gupta GP, Sahu SP (2023) Hybrid meta-heuristic based feature selection mechanism for cyber-attack detection in IoT-enabled networks. Procedia Comput Sci 218:318–327. https://doi.org/10.1016/j.procs.2023.01.014

10. SaiSindhuTheja R, Shyam GK (2021) An efficient metaheuristic algorithm based feature selection and recurrent neural network for DoS attack detection in cloud computing environment. Appl Soft Comput J 100:106997. https://doi.org/10.1016/j.asoc.2020
11 Manimurugan S, Majdi AQ, Mohmmed M, Narmatha C, Varatharajan R (2020) Intrusion detection in networks using crow search optimization algorithm with adaptive neuro-fuzzy inference system. Microprocessors Microsyst. https://doi.org/10.1016/j.micpro.2020.103261
12. Keserwani PK, Govil MC, Shubhakar E (2020) An optimal intrusion detection system using GWO-CSA-DSAE model. Cyber-Phys Syst. https://doi.org/10.1080/23335777.2020.1811383
13. Al Shorman A, Faris H, Aljarah I (2020) Unsupervised intelligent system based on one class support vector machine and Grey Wolf optimization for IoT botnet detection. J Ambient Intell Human Comput 11:2809–2825
14. Alzaqebah A, Aljarah I, Al-Kadi O, Damaševičius R (2022) A modified grey wolf optimization algorithm for an intrusion detection system. Mathematics 10(6):999. https://doi.org/10.3390/math10060999
15. Kumar P, Gupta GP, Tripathi R (2021) Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for IoT networks. Arab J Sci Eng 46:3749–3778. https://doi.org/10.1007/s13369-020-05181-3
16. Kundu R, Mallipeddi R (2022) HFMOEA: a hybrid framework for multi-objective feature selection. J Comput Design Eng 9(3):949–965. https://doi.org/10.1093/jcde/qwac040
17. Thaseen IS, Kumar CA, Ahmad A (2019) Integrated intrusion detection model using Chi-square feature selection and ensemble of classifiers. Arab J Sci Eng 44:3357–3368. https://doi.org/10.1007/s13369-018-3507-5
18. Dhanabal L, Shantharajah SP (2015) A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. Int J Adv Res Comput Commun Eng 4(6):446–452
19. Al-Khassawneh YA, An investigation of the Intrusion detection system for the NSL-KDD dataset using machine-learning algorithms. In: 2023 IEEE International Conference on Electro Information Technology (eIT), Romeoville, IL, USA, 2023, pp. 518–523, https://doi.org/10.1109/eIT57321.2023.10187360.
20. Alazzam H, Sharieh A, Sabri KE (2020) A feature selection algorithm for intrusion detection system based on Pigeon inspired optimizer. Exp Syst Appl 148:113249. https://doi.org/10.1016/j.eswa.2020.113249
21. Shirvani MH, Amirsoleimani N, Salimpour S, Azab A, Multi-criteria task scheduling in distributed systems based on fuzzy TOPSIS. In: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, Canada, 2017, pp. 1–4, https://doi.org/10.1109/CCECE.2017.7946721.
22. Aminizadeh S, Heidari A et al (2023) The applications of machine learning techniques in medical data processing based on distributed computing and the Internet of Things. Comput Methods Programs Biomed 241:107745. https://doi.org/10.1016/j.cmpb.2023.107745
23. Abedpour K, Hosseini Shirvani M, Abedpour E (2024) A genetic-based clustering algorithm for efficient resource allocating of IoT applications in layered fog heterogeneous platforms. Cluster Comput 27:1313–1331. https://doi.org/10.1007/s10586-023-04005-x
24. JavadianKootanaee A, Poor Aghajan AA, Hosseini Shirvani M (2021) A hybrid model based on machine learning and genetic algorithm for detecting fraud in financial statements. J Optim Ind Eng. 14(2):169–186
25. Khaledian N, Mardukhi F (2022) CFMT: a collaborative filtering approach based on the nonnegative matrix factorization technique and trust relationships. J Ambient Intell Human Comput 13:2667–2683. https://doi.org/10.1007/s12652-021-03368-6
20 Koroniotis N, Moustafa N, Sitnikova E, Turnbull B (2019) towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Gener Comput Syst 100:779–796
21 Moustafa N (2021) a new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. Sustain Cities Soc 72:102994
22 Zar JH (2005) Spearman rank correlation. Encyclop Biostat. https://doi.org/10.1002/0470011815.b2a15150
29. Norozpour S, Darbandi M, Proposing New Method for Clustering and Optimizing Energy Consumption in WSN. Talent Development & Excellence, 2020;12.

30. Darbandi M, et al. Prediction and Estimation of Next Demands of Cloud Users based on their Comments in CRM and Previous usages. in 2018 International Conference on Communication, Computing and Internet of Things (IC3IoT). 2018. IEEE.
31. Darbandi M, Kalman filtering for estimation and prediction servers with lower traffic loads for transferring high-level processes in cloud computing. Published by HCTL International Journal of Technology Innovations and Research,(ISSN: 2321–1814), 2017;23(1):10–20.
32. Weisberg S (2001) Yeo-Johnson Power Transformations; University ofMinnesota, StatisticsArc: Minnesota. MN, USA
33. Zouhri H, Idri A, Ratnani A (2024) Evaluating the impact of filter-based feature selection in intrusion detection systems. Int J Inf Secur 23:759–785. https://doi.org/10.1007/s10207-023-00767-y
34. Al-Yaseen WL, Idrees AK, Almasoudy FH (2022) Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system. Pattern Recognit 132:108912. https://doi.org/10.1016/j.patcog.2022.108912
35. Seifhosseini S, Shirvani MH, Ramzanpoor Y (2024) Multi-objective cost-aware bag-of-tasks scheduling optimization model for IoT applications running on heterogeneous fog environment. Comput Netw 240:110161. https://doi.org/10.1016/j.comnet.2023.110161
36. Hosseini Shirvani M, Ramzanpoor Y (2023) Multi-objective QoS-aware optimization for deployment of IoT applications on cloud and fog computing infrastructure. Neural Comput Applic 35:19581–19626. https://doi.org/10.1007/s00521-023-08759-8
37. Sahu SK, Sarangi S, Jena SK, A detail analysis on intrusion detection datasets. In: 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 2014, pp. 1348-1353, https://doi.org/10.1109/IAdCC.2014.6779523.
38. Heidari A, Shishehlou H et al (2024) A reliable method for data aggregation on the industrial internet of things using a hybrid optimization algorithm and density correlation degree. Cluster Comput 27:7521–7539. https://doi.org/10.1007/s10586-024-04351-4
39. Hwang CL, Yoon K (1981) Multiple Attribute Decision Making: Methods and Applications. Springer-Verlag, New York
40. https://pypi.org/project/pandas/ [visited 9/7/2024].
44 Kumar A, Kaur A, Singh P, Driss M, Boulila W (2023) Efficient multiclass classification using feature selection in high-dimensional datasets. Electronics. https://doi.org/10.3390/electronics12102290
43 Hosseini Shirvani M, Rahmani AM, Sahafi A (2017) An iterative mathematical decision model for cloud migration: A cost and security risk approach. Softw Pract Exp 48(3):449–485. https://doi.org/10.1002/spe.2528
43. https://www.kaggle.com/datasets/hassan06/nslkdd [visited 9/7/2024].
44. https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15 [visited 9/7/2024].
45. Jain D, Singh V (2018) Feature selection and classification systems for chronic disease prediction: a review. Egypt Inf J 19:179–189. https://doi.org/10.1016/j.eij.2018.03.002
46. Gupta C, Kumar A, Jain NK (2023) A detailed analysis on intrusion detection systems, datasets, and challenges. In: Chakraborty, B., Biswas, A., Chakrabarti, A. (eds) Advances in Data Science and Computing Technologies. ADSC 2022. Lecture Notes in Electrical Engineering, vol 1056. Springer, Singapore. https://doi.org/10.1007/978-981-99-3656-4_26.
47. Hosseini Shirvani M (2020) A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. Eng Appl Artif Intell. https://doi.org/10.1016/j.engappai.2020.103501