

Enhancing IoT Security: A Novel Feature Engineering Approach for ML-Based Intrusion Detection Systems

Afsaneh Mahanipour
Department of Computer Science
University of Kentucky
Lexington, KY, USA
ama654@uky.edu

Hana Khamfroush
Department of Computer Science
University of Kentucky
Lexington, KY, USA
khamfroush@cs.uky.edu

Abstract—The integration of Internet of Things (IoT) applications in our daily lives has led to a surge in data traffic, posing significant security challenges. IoT applications using cloud and edge computing are at higher risk of cyberattacks because of the expanded attack surface from distributed edge and cloud services, the vulnerability of IoT devices, and challenges in managing security across interconnected systems leading to oversights. This led to the rise of ML-based solutions for intrusion detection systems (IDSs), which have proven effective in enhancing network security and defending against diverse threats. However, ML-based IDS in IoT systems encounters challenges, particularly from noisy, redundant, and irrelevant features in varied IoT datasets, potentially impacting its performance. Therefore, reducing such features becomes crucial to enhance system performance and minimize computational costs. This paper focuses on improving the effectiveness of ML-based IDS at the edge level by introducing a novel method to find a balanced trade-off between cost and accuracy through the creation of informative features in a two-tier edge-user IoT environment. A hybrid Binary Quantum-inspired Artificial Bee Colony and Genetic Programming algorithm is utilized for this purpose. Three IoT intrusion detection datasets, namely NSL-KDD, UNSW-NB15, and BoT-IoT, are used for the evaluation of the proposed approach. Performance analysis is conducted using various evaluation metrics such as accuracy, sensitivity, specificity, and False Positive Rate (FPR) are employed, while the cost of the IDS system is assessed based on computational time. The results are compared with existing methods in the literature, revealing that the IDS performance can be enhanced with fewer features, consequently reducing computational time, through the proposed method. This offers a better performance-cost trade-off for the IDS system.

Index Terms—Binary Quantum-inspired Artificial Bee Colony Algorithm, Feature Construction, Feature Selection, Genetic Programming, Intrusion Detection Systems

I. INTRODUCTION

In the era of Internet of Things (IoT) applications and widespread internet usage, security has become a major and growing concern. The interconnected nature of IoT devices has expanded the attack surface, providing numerous entry points for malicious actors. The increasing volume of data exchanged between these devices poses a significant target for cybercriminals, raising the potential impact of security breaches across various aspects of daily life, from smart homes to industrial systems. By the end of 2024, it is anticipated that there will be 83 billion

IoT devices, utilized across various domains such as intelligent transportation, smart healthcare, and others, contributing to the development of smart cities [1].

This surge in IoT adoption has also led to challenges in data management and processing. The data collected by IoT devices has been transmitted to a distant cloud server for further processing. However, this setup poses challenges due to the significant distance between the devices and the server, resulting in delays that are not conducive to the time-sensitive nature of IoT applications. Therefore, with the introduction of an edge computing framework, data is sent to edge servers located closer to the IoT devices in most applications. This reduces latency, but it's important to note that these edge servers still face limitations in terms of storage and computational power compared to cloud servers [2].

Moreover, within IoT frameworks, numerous devices interact through web application interfaces, necessitating robust authentication and encryption methods to ensure secure communication. In this context, intrusion detection systems (IDSs) play a crucial role in monitoring and identifying attacks. IDSs are broadly categorized into two types based on their discovery methods: signature-based IDS and anomaly-based IDS. The signature-based method relies on pre-stored rules that represent specific attack types. Consequently, any attack not included in the pre-stored rules will go undetected. Conversely, anomaly-based IDS exhibits proficiency in efficiently detecting zero-day (unknown) attacks [3].

Lately, the application of machine learning (ML) techniques has become increasingly prevalent in the field of intrusion detection within IoT IDSs. However, IoT devices exhibit differences in their hardware attributes, functions, and computational capabilities to generate features. When IoT devices transmit their data to an edge server, some features may be noisy, irrelevant or redundant. These characteristics significantly affect the performance of IDSs [1]. Selecting informative features is pivotal in ML-based methods for enhancing IDS accuracy by eliminating non-informative features and reducing complexity costs through dataset size reduction. While some studies, such as [4, 5], have employed feature selection (FS) techniques for this

purpose, the exploration of constructing new high-level features for IDSs remains unaddressed in existing literature. This paper proposes a novel method to find a balanced trade-off between costs and accuracy by constructing new informative features for ML-based IDSs within the diverse IoT environment.

The search space for feature construction is extensive, as informative features must be combined with appropriate operators to create distinctive new features. Therefore, selecting prominent features for this purpose can be beneficial. To achieve this, the binary quantum-inspired artificial bee colony (BQABC) algorithm [6] is used for selecting informative features. BQABC has better convergence rate and exploration capability to prevent trapping in a local optima compared to other binary optimization algorithms such as binary quantum-inspired particle swarm optimization (BQIPSO) and binary quantum-inspired evolutionary algorithm (BQIEAo). Subsequently, genetic programming (GP) [7], a domain-independent problem-solving approach, is utilized to construct high-level features for more accurate intrusion detection with less computation cost. The main contributions of the proposed method can be summarized in the following manner:

- Utilizing feature construction method for ML-based IDSs for the first time
- Proposing a novel feature engineering method by integrating FS based on BQABC algorithm for the identification of important features and FC based on GP for constructing new feature to create a new dataset
- Evaluating the performance of the proposed method in terms of accuracy, sensitivity, specificity, False Positive Rate (FPR), number of features, and computational time
- Comparing the proposed method with seven other methods in the literature by employing various evaluation metrics and obtaining better results such as improving accuracy by approximately 8% and reducing computational time by an average of 2948 s across all three datasets

II. BACKGROUND AND RELATED WORKS

A. Related Works

Supervised ML classification algorithms use training data to establish a functional relationship between input features and class values. The trained classifier then predicts class values for query instances. However, the dataset may contain redundant, noisy, or irrelevant features, which can harm classification performance. To address this, feature engineering techniques like feature selection (FS), and feature construction (FC) are employed to enhance feature quality and boost classifier accuracy. FC creates new features through functional expressions to improve performance and reveal hidden relationships, while FS selects informative features and reduces the total number by eliminating non-informative ones [8].

While FS methods have been used in ML-based IDSs, FC techniques have not been applied. Consequently, this section investigates previous research studies that have employed various FS methods to enhance the performance of ML-based IDSs. FS methods can be mainly categorized into filter and wrapper methods. Filter methods operate independently of the

learning algorithms and rank features based on their inherent characteristics like information theory, mutual information, or correlation criteria [3]. For example, in [9], the features were ranked by combining their statistical importance through Standard Deviation and the Difference of Mean and Median. In another study [10], information gain was used as FS method.

In contrast, wrapper methods employ learning algorithms to assess the quality of selected features, making them more accurate than filter methods. For instance, in [11], Genetic algorithm with Random Forest-based fitness function was used to select informative features. Similarly, in [4], differential evolution algorithm was used to select the most suitable features and subsequently assessed the selected features using the extreme learning machine classifier. In the paper by Kareem et al. [5], they employed Gorilla Troops Optimizer as a FS algorithm and augmented its exploitation capability by integrating the bird swarms algorithm. In another study [12], the authors employed the Tabu Search algorithm in conjunction with a random forest classifier to identify the optimal subset of features for IDSs.

Reviews indicate the need for more optimal solutions due to low classification accuracy in existing methods. Additionally, FC methods have not been employed as a pre-processing step in IDSs. Therefore, this paper proposes a hybrid feature engineering method to find a balanced trade-off between computational cost and accuracy.

B. Binary Quantum-inspired Artificial Bee Colony

Binary Quantum-inspired Artificial Bee Colony (BQABC) is proposed by Barani and Nezamabadi-pour in 2017 [6]. It combines Artificial Bee Colony (ABC)'s main structure with quantum computing principles, offering high exploration capability and robustness for binary optimization problems.

In the BQABC, some concepts of quantum computing like quantum bits and quantum gates are applied in the main structure of ABC algorithm to define the position of food sources and their updating process. The pseudocode of the BQABC algorithm is given in Algorithm 1. You can read more details about this algorithm in [6].

C. Genetic Programming (GP)

Genetic Programming (GP) was introduced for the first time by John Koza [7]. This algorithm used genetic algorithm as a process to evolve mathematical functions, but chromosomes are encoded with tree structure. The leaves of trees can be selected from the variables of the problem, and internal nodes can be selected from predefined mathematical operators. GP is a population-based evolutionary algorithm that tries to find an optimum function for the desired problem. For this purpose, it generates a number of chromosomes with different sizes, and then selects the best one based on their fitness values. The GP search procedure can be narrated by several main steps:

- 1) Initializing a random population of individual chromosomes using variables and operators.
- 2) Iterating the following sub-steps until reaching a stopping criterion:

Algorithm 1 Pseudocode of the BQABC algorithm

Input: The number of population and Maximum iterations (termination condition)

Output: Best food source

```
1: Initialize quantum food sources ( $Q(t)$ ) randomly, a set of
   best food sources  $FB(t) = \{\}$ , a set of current food sources
    $FW(t) = \{\}$ , and  $t = 0$ 
2: while not termination condition do
3:   Observe  $Q(t)$  and make  $FW(t)$ 
4:   Calculate fitness values of  $F_i(t) \in FW(t)$ , by a desired
   fitness function ( $fit(F_i)$ )
5:   Update  $FB(t)$ 
6:   for each employed bee  $i$  do
7:     Generate a new quantum food source  $q'_i$  in the
     neighborhood of  $q_i$  using Equations (12,13,14) in [6].
8:     Observe  $q'_i$  and make  $F'_i$ 
9:     Calculate fitness value of  $F'_i$ 
10:    if  $fit(F'_i) > fit(F_i)$  then
11:       $q_i = q'_i$ 
12:       $F_i = F'_i$ 
13:    end if
14:  end for
15:  for each onlooker bee  $j$  do
16:    Calculate the probability of food sources using Eq.
    4 in [6].
17:    Select a quantum food source  $q_j$  based on probability
    values
18:    Generate a new quantum food source  $q'_j$  in the
    neighborhood of  $q_j$  using Equations (12,13,14) in [6].
19:    Observe  $q'_j$  and make  $F'_j$ 
20:    Calculate fitness value of  $F'_j$ 
21:    if  $fit(F'_j) > fit(F_j)$  then
22:       $q_j = q'_j$ 
23:       $F_j = F'_j$ 
24:    end if
25:  end for
26:  Determine abandoned food source and replace it with a
  new quantum food source for the scout bee
27:  Memorize the best food source found so far
28:   $t = t + 1$ 
29: end while
```

- a) Evaluation: calculating fitness value of each chromosome by an appropriate fitness function.
- b) Selection: choosing one or more chromosomes of the population by a selection approach to participate in the next sub-step.
- c) Evolution: generating new chromosomes and developing a new population by applying genetic operators including: reproduction, crossover, and mutation on the selected chromosomes.

3) Returning the best chromosome with the maximum fitness value as the optimum solution.

III. PROPOSED METHOD

In this section, the proposed feature engineering method for intrusion detection is explained. This method utilizes both FS and FC techniques to provide informative features for precise classification of network attacks, while also finding the trade-off between accuracy and computational cost. To achieve this, the BQABC nature-inspired algorithm is employed to evaluate dataset features and eliminate non-informative ones, thereby reducing the search space and improving the quality of the feature set. For example, consider a dataset $D = \{X, Y\} := \{(x_n, y_n)\}_{n=1}^N$, consisting of N samples and a feature set $F = \{f_1, f_2, \dots, f_m\}$. Here, $x_n = (x_{n1}, x_{n2}, \dots, x_{nm})$ represents a sample vector, and $Y = (y_1, y_2, \dots, y_N)$ denotes the class label vector of N samples, where $y_n \in \{1, \dots, ClassLabel\}$. $F = \{f_1, f_2, \dots, f_m\}$ refers the original feature set used by BQABC to select informative features $Selected - F = \{f_1, f_2, \dots, f_s\}$. Subsequently, the selected features are inputted into the GP algorithm, a type of FC algorithm, to generate a new, higher-level, and more distinctive feature $\{f_c\}$. This newly constructed feature is then added to the selected feature subset, resulting in an augmented feature set $Augmented - F = \{f_1, f_2, \dots, f_s, f_c\}$. Finally, this augmented feature set is fed into a Machine Learning (ML) classifier to detect security attacks.

System Architecture: In this paper, intrusion attack detection is performed at the edge level. Data collected by the IoT devices are sent to their closest edge server for further processing including intrusion detection. As previously stated, the distance (D) between IoT devices and the cloud server is significantly greater than the distance (d) between IoT devices and the nearest edge server. Therefore, by transmitting data to the edge server, latency is reduced, making it more suitable for time-sensitive IoT applications. However, edge servers have inherent limitations in terms of storage and computational power compared to cloud servers. As illustrated in Fig. 1, our focus is solely on detecting attacks on a single edge server, assuming that each edge server is tasked to perform intrusion detection for its assigned set of IoT devices.

FS phase: The BQABC is a population-based method, and its procedure begins by initializing a random population. Each agent in this population is represented by a binary string, with a length exactly matching the number of original features in a dataset. The i th binary solution at the t th iteration of this algorithm can be represented as $Z_i(t) = [z_i^1(t), z_i^2(t), \dots, z_i^m(t)]$, where $z_i^j(t) \in \{0, 1\}$. In these strings, “1” indicates that the corresponding feature is selected, while “0” indicates it is not selected. Subsequently, these agents (primary solutions) are displaced, combined and evolved during the iterations. To achieve this, at each iteration, a learning algorithm’s feedback is utilized to evaluate and score the agents. In other words, the learning algorithm serves as a fitness function, with its accuracy considered as a fitness value for the agents. Ultimately, the best-performing agent is identified as the optimal feature subset.

FC phase: In this step, the GP algorithm is employed to generate a new informative feature. Unlike most other population-based methods, the GP algorithm represents individuals as trees, which serves as a robust representation for mathematical

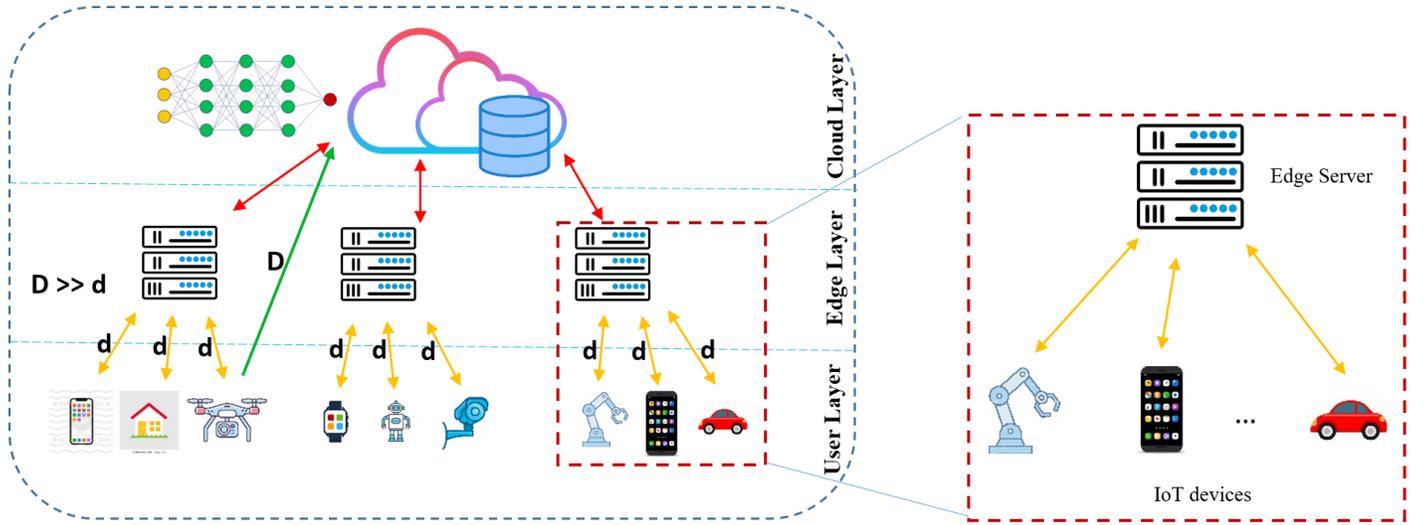


Fig. 1: System Architecture.

expressions. The process begins by initializing the population, wherein the selected features obtained from the previous phase $Selected - F = \{f_1, f_2, \dots, f_s\}$ (FS phase) are combined with mathematical operations such as $+$, $-$, \times , \sin , and \cos . Each tree, or newly created feature, is then evaluated using a classifier, and the accuracy of that classifier is assigned to the corresponding constructed feature as a fitness value. The process continues until the maximum number of allowed generation is reached, during which trees are reproduced using crossover and mutation operators to update the population. Upon reaching the termination condition, the best individual is selected as the high-level constructed feature.

After these two steps, the best constructed feature from the FC step $\{f_c\}$ is added to the best selected features that are obtained in the FS step ($Selected - F = \{f_1, f_2, \dots, f_s\}$). This augmented feature set ($Augmented - F = \{f_1, f_2, \dots, f_s, f_c\}$) is used for training a classifier to detect network attacks and calculate its accuracy as a measure of the IDS's performance. The pseudocode of the proposed method is given in Algorithm 2, and this process is illustrated in Fig. 2.

IV. EXPERIMENTS AND RESULTS

A. Dataset

The proposed method is tested on three benchmark datasets: NSL-KDD [13], UNSW-NB 15 [14], and Bot-IoT [15]. The NSL-KDD dataset, an improved version of KDD99, comprises 41 features with 125,973 instances in the training set and 22,544 instances in the test set. The UNSW-NB 15 dataset includes 49 features created using the Argus tool. It is divided into a training set containing 175,341 samples and a testing set comprising 82,332 samples, covering various attack types and standard records. The Bot-IoT dataset is created to mimic a real-world scenario. This paper utilizes 5% of the complete dataset, which comprises over 3.6 million instances [5]. The characteristics of these datasets are indicated in Table I.

Algorithm 2 Pseudocode of the proposed method

Input: The population size for BQABC (S), Number of original features (m), The population size for GP (S'), The maximum number of iterations, Predefined mathematical operations

Output: The best agent representing the best selected features and the best program representing the best constructed feature

- 1: Initialize the population of BQABC randomly
 - 2: **while** reaching the stopping criteria or maximum iteration **do**
 - 3: Evaluate agents by a fitness function
 - 4: Generate new quantum food sources and update agents' positions based on BQABC algorithm (Algorithm 1)
 - 5: **end while**
 - 6: Save the best agent representing the best selected features
 - 7: Extract the best selected features from the best agent
 - 8: Initialize the population of GP randomly by using the best selected features as variables and predefined mathematical operations
 - 9: **while** reaching the stopping criteria or maximum iteration **do**
 - 10: Evaluate programs/constructed features by a fitness function
 - 11: Select one or two programs by a selection approach
 - 12: Apply genetic operators including: reproduction, crossover, and mutation on the selected chromosomes
 - 13: **end while**
 - 14: Save the best program representing the best constructed feature
 - 15: Add the best constructed feature to the best selected ones
-

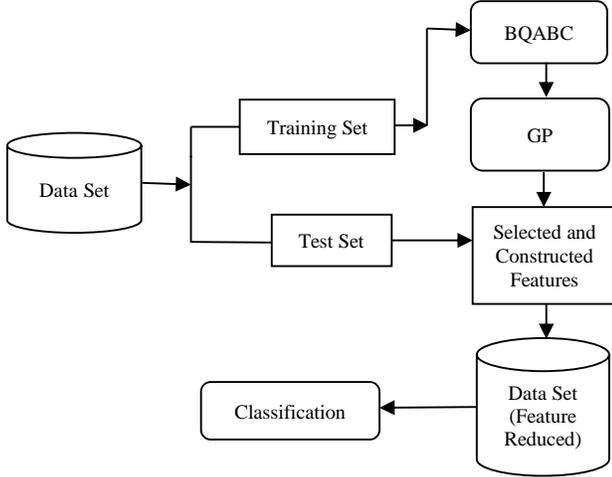


Fig. 2: Overview of the proposed method.

TABLE I: Characteristics of the datasets

Criteria (↓)/Dataset name (→)	NSL-KDD	UNSW-NB15	BoT-IoT
Number of features	41	49	43
Number of attack categories	4	9	4
Number of classes	5	10	5
Number of total instances	148517	257673	3668522
Number of instances in training set	125973	175341	2934817
Number of instances in test set	22544	82332	733705

B. Evaluation Measure

In wrapper-based FS and FC methods, performance of a classifier is used to evaluate and score selected feature subsets or constructed features. In this paper, K-nearest neighbor (KNN) classifier is used as a fitness function in BQABC and GP algorithm. The performance metrics extracted from the confusion matrix and employed in this study are accuracy, sensitivity (recall), specificity, and the False Positive Rate (FPR). These evaluation metrics are calculated using equations (1) to (4).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

$$Sensitivity(recall) = \frac{TP}{TP + FN} \quad (2)$$

$$Specificity = \frac{TN}{TN + FP} \quad (3)$$

$$FPR = \frac{FP}{FP + TN} \quad (4)$$

where TP , TN , FP , and FN denote true positive, true negative, false positive, and false negative, respectively.

C. Parameter Setting

The KNN Learning algorithm is employed as the classifier. Compared to more complex learning algorithms such as neural networks, KNN is straightforward yet performs well, often competing with other complex classifiers. Given the limitations of edge servers previously discussed, a simple classifier like

KNN is used in this work. According to [5], the value of K in the KNN classifier, denoting the number of nearest neighbors, is set to 5 for result comparison. In BQABC, the minimum and maximum values of the rotation angle which is indicated by θ are 0.001π and 0.05π respectively. The total number of agents in BQABC and GP is set to 50, and maximum number of iterations is 100. Each of these methods is executed 30 times, and the average evaluation values are computed based on these 30 runs. It should be mentioned that all these values are chosen based on trial and error. All these parameters are demonstrated in Table II.

TABLE II: Parameters of the proposed method

Parameter	Parameter value
Population size of BQABC	50
Population size of GP	50
Maximum number of iterations	100
Number of runs	30
Number of neighbors in KNN	$K = 5$
Maximum value of the rotation angle	$\theta = 0.05\pi$
Minimum value of the rotation angle	$\theta = 0.001\pi$
Internal node's functions	$+, -, \times, \sin, \cos$

Data pre-processing: Also, the data is initially pre-processed to ensure uniform formatting. This involves a two-step process: first, non-numeric values are converted to numeric values, referred to as numericalization. Following this, a normalization process is applied to bring all numerical columns onto a common scale range. Some variables, when measured on different scales, can introduce bias into the model. Given that we are working with NSL-KDD, UNSW-NB15, and BoT-IoT as benchmark datasets, it has been observed that each feature exhibits distinct ranges. Therefore, min-max normalization is chosen for consistency. This approach involves normalizing the features by subtracting the mean from each feature and then dividing it by its standard deviation. Min-max normalization scales the values to fall within the range of $[0, 1]$.

D. Results and Analysis

In this section, we evaluate the effectiveness of the proposed hybrid feature selection and construction method in identifying the optimal feature subset and constructed feature by comparing it to various other metaheuristic feature selection algorithms, such as CS-PSO [16], HHO [17], MVO [18], HGS [19], BSA [20], GTO [21], and GTO-BSA [5] which are described below. The experiments make use of three intrusion detection datasets: NSL-KDD, UNSW-NB15, and BoT-IoT, representing a good combination of zero-day security threats.

- Cuckoo search-Particle swarm optimization (CS-PSO): It combines two algorithms, cuckoo search and particle swarm optimization, while integrating the concepts of local best and global best. Then informative features are chosen based on their fitness values to classify all the attacks accurately .
- Harris Hawks Optimize (HHO): It is a nature-inspired algorithm that mimics the cooperative behavior and hunting style of Harris's hawks in the wild, known as the 'surprise

pounce’. It is used to remove irrelevant features to improve the performance of IDSs.

- **Multi-Verse Optimizer (MVO):** This algorithm is rooted in three cosmological principles: white holes, black holes, and wormholes.
- **Hunger Games Search (HGS):** It is crafted to align with the hunger-driven behaviors and choices observed in animals.
- **Bird Swarm Algorithm (BSA):** It is derived from swarm intelligence observed in bird swarms, encompassing three primary behaviors in birds: foraging, vigilance, and flight.
- **Gorilla Troops Optimizer (GTO):** This algorithm mathematically models the collective life of gorillas and introduces novel mechanisms for both exploration and exploitation.
- **Gorilla Troops Optimizer based on Bird Swarm Algorithm (GTO-BSA):** This algorithm utilizes BSA to enhance the exploitation performance of GTO due to its strong capability in locating feasible regions with optimal solutions. Consequently, this contributes to an improved final output quality and enhances the performance of IDSs.

Comparison with state of the art methods: Tables III, IV, and V present the experimental results for NSL-KDD, UNSW-NB15, and BoT-IoT, respectively. The results indicate that, in nearly all evaluation metrics, the proposed hybrid feature selection and feature construction technique outperforms recent wrapper-based feature selection techniques with KNN-based IDS for all three intrusion detection datasets. In the case of the NSL-KDD dataset, the proposed approach attained an accuracy of 0.9889 with a reduced dataset containing 11 features. This reflects an approximate increase in accuracy of 0.033 to 0.0449 with the proposed hybrid method when applied to a KNN-based IDS using the NSL-KDD dataset. Meanwhile, on the UNSW-NB15 dataset, the proposed approach achieved an accuracy of 0.9022 while reducing the feature set by approximately 78.36%. As a result, an estimated increase in accuracy of approximately 0.1921 to 0.2696 is observed with the proposed method. Moving on to the BoT-IoT dataset, the proposed method achieved an accuracy of 0.9849 using a reduced feature subset consisting of about 10 features out of the original 43. This resulted in an estimated accuracy increase of around 0.0317 to 0.0525. However, the proposed method selects and constructs more features for this dataset compared to the other FS methods being compared.

A high False Positive Rate (FPR) can trigger false alarms, potentially causing the intrusion detection system to misclassify normal network traffic as malicious. With the proposed approach, it achieves promising scores for sensitivity (True Positive Rate), specificity (True Negative Rate), and FPR evaluation metrics across all three datasets. This method concurrently enhances both the True Positive Rate and True Negative Rate while reducing the FPR. Compared to other methods, the NSL-KDD dataset exhibits an increase in sensitivity of approximately 0.0561 to 0.0809 and an increase in specificity of around 0.0121 to 0.0169. These improvements result in a sensitivity of 0.9703 and a specificity of 0.9876 when applying the proposed hybrid technique. Similarly, for the UNSW-NB15 dataset, there is an

increase in sensitivity ranging from about 0.133 to 0.2656 and an improvement in specificity ranging from approximately 0.0023 to 0.0779, culminating in a sensitivity of 0.9483 and a specificity of 0.8806. Nonetheless, in the case of the BoT-IoT dataset, the HHO method achieved a higher sensitivity of 0.9992, while the sensitivity of the proposed method was 0.9979. On the other hand, the proposed method obtained the highest specificity of 0.9927, whereas HHO’s specificity was 0.5111. Furthermore, the proposed method has demonstrated superior performance in terms of achieving the lowest FPR across all three datasets.

Computational time analysis: Table VI illustrates the average computational time (s) across 30 repetitions for the compared methods. Given our focus on the performance-cost trade-off, it is essential to simultaneously consider Table VI alongside Tables III, IV, and V for a comprehensive understanding. By comparing table VI and table III, it is clear that the proposed method outperforms the overall best-performing algorithm (GTO-BSA) in terms of speed by a factor of 7 while also enhancing accuracy from 0.9559 to 0.9889. This comparison highlights that the proposed method offers a balanced trade-off between computational cost and other performance metrics for almost all datasets. For example, by considering Table VI and Table V, it becomes apparent that the proposed method outperforms the HHO method in computational cost by a factor of 1.5. Furthermore, the sensitivity remains nearly identical (0.9979), while other performance metrics exhibit improvement.

TABLE III: Comparison of the proposed method with existing methods in the literature on the NSL-KDD dataset.

Method	(#) Features	Accuracy	Sensitivity	Specificity	FPR
CS-PSO	14.875	0.9501	0.8894	0.9755	0.0245
HHO	19.625	0.9544	0.9059	0.9733	0.0267
MVO	16.5	0.9531	0.8981	0.9749	0.0251
HGS	18.625	0.9479	0.8945	0.9743	0.0257
BSA	21.125	0.9440	0.9003	0.9707	0.0293
GTO	18.5	0.9543	0.9032	0.9738	0.0262
GTO-BSA	14.75	0.9559	0.9142	0.9736	0.0264
Proposed method	11	0.9889	0.9703	0.9876	0.0124

TABLE IV: Comparison of the proposed method with existing methods in the literature on the UNSW-NB15 dataset.

Method	(#) Features	Accuracy	Sensitivity	Specificity	FPR
CS-PSO	18.125	0.6692	0.7730	0.8663	0.1337
HHO	12	0.7064	0.7865	0.8222	0.1778
MVO	16.5	0.6979	0.7807	0.8783	0.1217
HGS	18.75	0.6326	0.6827	0.8401	0.1599
BSA	14.875	0.6543	0.7519	0.8675	0.1325
GTO	12.625	0.7072	0.7788	0.8027	0.1973
GTO-BSA	16.625	0.7101	0.8153	0.8770	0.1230
Proposed method	10.6	0.9022	0.9483	0.8806	0.1194

Non-parametric statistical analysis: The proposed approach enhances IDS performance in most cases by carefully choosing and creating informative features from the original feature set. To statistically validate the achieved results, the nonparametric Wilcoxon signed-rank test is utilized for all the performance measures. In Table VII, the exact p-values are presented, confirming significant differences between the proposed hybrid

TABLE V: Comparison of the proposed method with existing methods in the literature on the BoT-IoT dataset.

Method	(#) Features	Accuracy	Sensitivity	Specificity	FPR
CS-PSO	3.2	0.9370	0.9648	0.9284	0.0716
HHO	2.133	0.9532	0.9992	0.5111	0.4889
MVO	2.8	0.9393	0.9670	0.8584	0.1416
HGS	3.6	0.9351	0.9620	0.9277	0.0723
BSA	3.4	0.9324	0.9512	0.6504	0.3496
GTO	2.4666	0.9479	0.9885	0.6500	0.3500
GTO-BSA	2.5333	0.9485	0.9928	0.9622	0.0378
Proposed method	10.33	0.9849	0.9979	0.9927	0.0073

TABLE VI: Comparison of the proposed method with existing methods in the literature in terms of computational time (s)

Method (\downarrow)/Dataset name (\rightarrow)	NSL-KDD	UNSW-NB15	BoT-IoT
CS-PSO	4604.31	80.89	71.09
HHO	12,476.16	146.24	144.71
MVO	5441.159	77.87	70.17
HGS	661.72	12.57	10.74
BSA	6515.84	74.88	68.97
GTO	9719.66	113.36	108.63
GTO-BSA	10,205.83	161.23	145.74
Proposed method	1441.02	133.55	92.50

method and the other compared algorithms. According to this test, any methods with p-values less than 0.05 are rejected. Notably, the proposed method rejects all of the comparative algorithms.

V. CONCLUSION AND FUTURE WORKS

In this paper, we introduce a novel feature engineering method to find a balance trade-off between cost and accuracy to enhance intrusion detection system (IDS) performance. This approach focuses on increasing detection accuracy while concurrently reducing false positive rates by identifying the most informative features that positively impact IDS performance. We evaluate the performance of the proposed feature engineering method using three IoT intrusion detection datasets: NSL-KDD, UNSW-NB15, and BoT-IoT, and compare it with other competitive algorithms. The results indicate achieved accuracies of 0.9889, 0.9022, and 0.9849 for the NSL-KDD, UNSW-NB15, and BoT-IoT datasets, respectively. As this study marks the initial exploration of feature construction in the context of intrusion detection, future research endeavors may extend this approach to construct multiple features, potentially replacing the total number of original features with the newly created ones. Furthermore, in future endeavors, our focus may shift towards detecting attacks within a distributed IoT environment, rather than concentrating solely on a single edge server.

ACKNOWLEDGEMENT

This work is funded by research grant provided by the National Science Foundation (NSF) under the grant number 1948387.

REFERENCES

[1] K. Albulayhi, Q. Abu Al-Haija, S. A. Alsuhibany, A. A. Jillepalli, M. Ashrafuzzaman, and F. T. Sheldon, "Iot in-

TABLE VII: Results of the Wilcoxon signed-rank test for the proposed method

Proposed method VS.	Exact P-value
CS-PSO	0.003906
HHO	0.007812
MVO	0.003906
HGS	0.003906
BSA	0.003906
GTO	0.003906
GTO-BSA	0.003906

trusion detection using machine learning with a novel high performing feature selection method," *Applied Sciences*, vol. 12, no. 10, p. 5015, 2022.

- [2] A. Mahanipour and H. Khamfroush, "Wrapper-based federated feature selection for iot environments," in *2023 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2023, pp. 214–219.
- [3] A. K. Mishra and S. Paliwal, "Mitigating cyber threats through integration of feature selection and stacking ensemble learning: the lgbm and random forest intrusion detection perspective," *Cluster Computing*, vol. 26, no. 4, pp. 2339–2350, 2023.
- [4] W. L. Al-Yaseen, A. K. Idrees, and F. H. Almasoudy, "Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system," *Pattern Recognition*, vol. 132, p. 108912, 2022.
- [5] S. S. Kareem, R. R. Mostafa, F. A. Hashim, and H. M. El-Bakry, "An effective feature selection model using hybrid metaheuristic algorithms for iot intrusion detection," *Sensors*, vol. 22, no. 4, p. 1396, 2022.
- [6] F. Barani and H. Nezamabadi-pour, "Bqiabc: a new quantum-inspired artificial bee colony algorithm for binary optimization problems," *Journal of AI and Data Mining*, vol. 6, no. 1, pp. 133–143, 2018.
- [7] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and computing*, vol. 4, pp. 87–112, 1994.
- [8] A. Mahanipour and H. Nezamabadi-Pour, "A multiple feature construction method based on gravitational search algorithm," *Expert Systems with Applications*, vol. 127, pp. 199–209, 2019.
- [9] A. Thakkar and R. Lohiya, "Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system," *Information Fusion*, vol. 90, pp. 353–363, 2023.
- [10] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on unsw-nb15 data set and the real time online dataset," *Cluster Computing*, vol. 23, pp. 1397–1418, 2020.
- [11] Z. Liu and Y. Shi, "A hybrid ids using ga-based feature selection method and random forest," *Int. J. Mach. Learn. Comput*, vol. 12, no. 02, pp. 43–50, 2022.
- [12] A. Nazir and R. A. Khan, "A novel combinatorial optimization based feature selection method for network intrusion

- detection,” *Computers & Security*, vol. 102, p. 102164, 2021.
- [13] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.
- [14] N. Moustafa and J. Slay, “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)*. IEEE, 2015, pp. 1–6.
- [15] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, “Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset,” *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.
- [16] P. Ghosh, A. Karmakar, J. Sharma, and S. Phadikar, “Cspso based intrusion detection system in cloud environment,” in *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 1*. Springer, 2019, pp. 261–269.
- [17] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, “Harris hawks optimization: Algorithm and applications,” *Future generation computer systems*, vol. 97, pp. 849–872, 2019.
- [18] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, “Multi-verse optimizer: a nature-inspired algorithm for global optimization,” *Neural Computing and Applications*, vol. 27, pp. 495–513, 2016.
- [19] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, “Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts,” *Expert Systems with Applications*, vol. 177, p. 114864, 2021.
- [20] X.-B. Meng, X. Z. Gao, L. Lu, Y. Liu, and H. Zhang, “A new bio-inspired optimisation algorithm: Bird swarm algorithm,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 4, pp. 673–687, 2016.
- [21] B. Abdollahzadeh, F. Soleimani Gharehchopogh, and S. Mirjalili, “Artificial gorilla troops optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems,” *International Journal of Intelligent Systems*, vol. 36, no. 10, pp. 5887–5958, 2021.