Uludağ University Journal of The Faculty of Engineering, Vol. 26, No. 1, 2021 RESEARCH

DOI: 10.17482/uumfd.747078

RULE GENERATION BASED ON MODIFIED CUTTLEFISH ALGORITHM FOR INTRUSION DETECTION SYSTEM

Adel Sabry EESA [*]
Sheren Sadiq [**]
Masoud Muhammed Hassan [*]
Zeynep ORMAN [***]

Received: 02.06.2020; revised: 25.01.2021; accepted: 25.01.2021

Abstract: Nowadays, with the rapid prevalence of networked machines and Internet technologies, intrusion detection systems are increasingly in demand. Consequently, numerous illicit activities by external and internal attackers need to be detected. Thus, earlier detection of such activities is necessary for protecting data and information. In this paper, we investigated the use of the Cuttlefish optimization algorithm as a new rule generation method for the classification task to deal with the intrusion detection problem. The effectiveness of the proposed method was tested using KDD Cup 99 dataset based on different evaluation methods. The obtained results were also compared with the results obtained by some classical well-known algorithms namely Decision Tree (DT), Naïve Bayes (NB), Support Vector Machine (SVM), and K-Nearest Neighborhood (K-NN). Our experimental results showed that the proposed method demonstrates a good classification performance and provides significantly preferable results when compared with the performance of other traditional algorithms. The proposed method produced 93.9%, 92.2%, and 94.7% in terms of precision, recall, and area under curve, respectively.

Keywords: Intrusion Detection System, Data Mining, Cuttlefish Algorithm, Classification, Rule Discovery

Saldırı Tespit Sistemi için Değiştirilmiş Mürekkep Balığı Algoritması Tabanlı Kural Üretimi

Oz: Günümüzde, ağa bağlı makinelerin ve Internet teknolojilerinin hızla yaygınlaşmasıyla, saldırı tespit sistemleri giderek daha fazla talep görmektedir. Buna bağlı olarak, dış ve iç saldırganların çok sayıda yasadışı faaliyetinin tespit edilmesi gerekmektedir. Bu nedenle, veri ve bilgilerin korunması için bu tür yasadışı faaliyetlerin erken tespiti gerekli ve önemlidir. Bu makalede, veri madenciliğinde saldırı tespit problemiyle başa çıkmak amacıyla Mürekkepbalığı Optimizasyon Algoritmasının yeni bir kural oluşturma yöntemi olarak kullanımı araştırılmıştır. Önerilen yöntemin etkinliği, farklı değerlendirme yöntemlerine dayalı olarak KDD Cup 99 veri seti kullanılarak test edilmiştir. Ayrıca, elde edilen sonuçlar Karar Ağacı, Naïve Bayes, Destek Vektör Makinesi ve K-En Yakın Komşu gibi bazı klasik iyi bilinen algoritmalar ile alınan sonuçlarla karşılaştırılmıştır. Deneysel sonuçlarımız, önerilen yöntemin iyi bir sınıflandırma performansı sergilediğini ve diğer geleneksel algoritmaların performansıyla karşılaştırıldığında önemli ölçüde tercih edilebilir sonuçlar verdiğini göstermektedir. Önerilen yöntem, hassasiyet, geri çağırma ve eğri altındaki alan açısından sırasıyla %93.9, %92.2 ve %94.7 değerlerini elde etmiştir.

Anahtar Kelimeler: Saldırı Tespit Sistemi, Veri Madenciliği, Mürekkepbalığı Algoritması, Sınıflandırma, Kural Keşfi

[*] University of Zakho, Department of Computer Science, P.O. Box 12, Duhok, Kurdistan, Iraq
[**] Duhok Polytechnic University, Department of Information Technology Management, 1006, Duhok, Kurdistan, Iraq
[***] Istanbul University-Cerrahpasa, Department of Computer Engineering, 34320, Avcilar, Istanbul, Turkey
Corresponding Author: Zeynep Orman (ormanz@istanbul.edu.tr)

1. INTRODUCTION

Extensive use of the Internet and data sharing on the web led the security of the networks to become a challenging issue (Duric, 2014; Zhang, 2020). An intrusion detection system (IDS) is one of the most important methods used to strengthen the security of the web and help the computer systems on how to deal with attacks (Khraisat, 2019). Intrusion is defined as an illicit attempt to access the computer systems, and IDS is a critical technology in terms of software and hardware that automates the procedure of monitoring and analyzing of illegal events. It is known as one of the best suitable methods to prevent and reveal such attacks (Jose, 2018; Vancea, 2014).

Knowledge Discovery in Databases (KDD) is defined as the operation of extracting patterns and models from large databases. Data mining is often used as a synonym for the KDD process, and it refers to the process of applying the discovery algorithm to the data (Schuh, 2019). One of the most important data mining techniques for IDSs is the rule discovery which tries to locate a collection of rules that can recognize the specific class from various groups of classes. Such a rule discovery tool is an essential phase for the classification tasks (Kiziloluk & Alatas, 2015; Patel & Buddhadev, 2015).

Many classification techniques are applied in the literature for the IDS problem, such as Support Vector Machine (Yinhui Li, 2012; Sumaiya Thaseen & Aswani Kumar, 2017; Zhao, 2010), Decision Tree (Eesa, 2015; Issa & Brifcani, 2011; Khraisat, 2018; Panigrahi & Borah, 2018), Naïve Bayes (Koc, 2012; Mukherjee & Sharma, 2012; Swarnkar & Hubballi, 2016), K-Nearest Neighborhood (Aburomman & Ibne Reaz, 2016; Li, 2014; Yang Li & Guo, 2007) and many more. In addition, evolutionary algorithms are widely used in IDS domain, including biology-inspired algorithms, such as genetic algorithm (Gauthama Raman, 2017; Hamamoto, 2018), swarm optimization(Ali & Jantan, 2011; Chung & Wahid, 2012; Kanaka Vardhini & Sitamahalakshmi, 2017), and ant colony optimization (Aghdam & Kabiri, 2016; Varma, 2016).

CFA was also successfully applied to generate and select features in the study of (Eesa, 2015; Eesa, 2017), the CFA were used in these studies as a feature selection while the DT algorithm was used as a classifier to measure the quality of the selected features. The results showed that the feature subset obtained by the CFA gave higher detection and higher accuracy rates with lower false alarm rates when compared with the obtained results using all features. The same method of the CFA was also considered by (Balasaraswathi, 2018) as a feature selection, but this time instead of using the random numbers, six different chaotic maps were used to acclimatize the CFA parameters. The randomness of the CFA was improved by using the chaotic random sequences Chaotic CFA, which improved the performance of CFA in terms of accuracy, detection rate, computation time and false alarm rate. In the above-mentioned studies, the operator in case 5 of the CFA was used to reduce the number of features by removing one feature at a time and the remaining features were evaluated by the DT classifier. Consequently, if the DT produced better results, this feature will be removed, and this process was repeated for each feature until all the features were tested. This feature selection strategy exists in the CFA was the main motivation behind using this algorithm for rule generation instead of features selection. In the proposed method, the CFA was used to generate several rules for each class, and in the rule pruning process, the case 5 was used to remove one rule with its corresponding feature at a time. In other words, if Rule (i) with its corresponding feature (i) is removed and better results obtained, then the Rule (i) will be removed and the next rule Rule(i+1) will be tested. This process is repeated until all rules are examined. To the best of our knowledge, this is the first attempt to use the CFA as a rule generation tool in this area of research. In this study, we aim to use the CFA as a rule generator for IDS problem. The generated rules are represented by several sets of Max and Min vectors for each feature. These rules will then be used to classify the test data to one of the five target classes exist in KDD Cup 99 dataset.

2. CUTTLEFISH OPTIMIZATION ALGORITHM (CFA)

Cuttlefish algorithm (CFA) is a new bio-inspired optimization method, which was proposed in 2013 by (Eesa, 2013) and was successfully used as a possible alternative tool for global optimization problems (Eesa, 2014), dimensionality reduction (Arshak & Eesa, 2018) and clustering problems (Eesa & Orman, 2019). This algorithm simulates the process of light reflection through the three skin layers of a cuttlefish, including iridophores, chromatophores, and leucophores. The interaction between these three layers through the six cases are shown in Figure 1 that allows the cuttlefish to produce complex patterns and colors.

There are two main processes considered for this algorithm. The first process is called the *reflection*, which mimics the light reflection, and the second process is called the *visibility* that simulates the matching patterns. These two processes are formulated in (1) to calculate the new solution.

$$new_p = reflection + visibility \tag{1}$$

The formulas for the interaction between the three layers of cells in six cases are described as follows:

For case 1 and 2:

$$reflection[j] = R * S_1\ [i].points[j] \tag{2}$$

$$visibility[j] = V * (Best\_points[j] - S_1[i].points[j]) \tag{3}$$

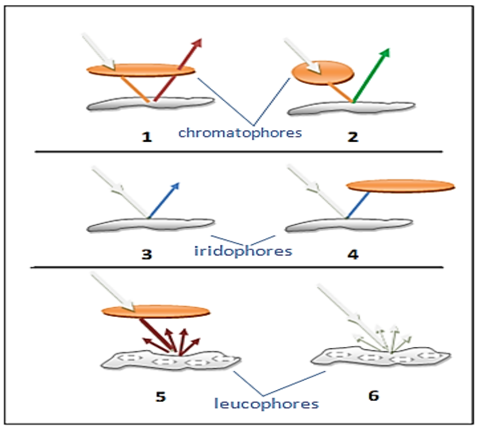

Figure 1:
Six cases of cells interaction used by the cuttlefish

where R and V are random variables with the values varying between (-1, 1), S_1 is a subset of the solutions, i is the i^{th} element in S_1, j is the j^{th} point in the element i, and *Best_points* denotes the best solution points. For these two cases, the value of R is generated when V is set to 1.

For case 3 and 4:

$$reflection[j] = R * Best\_points[j] \tag{4}$$

$$visibility[j] = V * (Best\_points[j] - S_2\ [i].points[j]) \tag{5}$$

where R is equal to 1, and V is generated randomly.

For case 5:

$$reflection[j] = R * Best\_points[j] \tag{6}$$

$$visibility[j] = V * (Best\_points[j] - AV\_best) \tag{7}$$

where *Best_points* is the optimal solution, and *AV_best* is the average of all optimal solutions. In this case, the value of *R* is generated, and *V* is set to 1.

For case 6:

$$new_P = random * (U - L) + L \tag{8}$$

where *random* is a random value between 0 and 1, *U* and *L* are the upper and lower limits of the problem domain. The algorithm divides the population into four subsets as S_1, S_2, S_3, and S_4. (2) and (3) in cases 1 and 2 are formulated to be used for the first subset of cells S_1 whereas (4) and (5) in cases 3 and 4, (6) and (7) in case 5 and (8) in case 6 are considered for S_2, S_3, and S_4, respectively. The main steps of CFA are expressed in Figure 2 as follows.

```
Initialize the population P with random solutions using Equation (8).
Compute and keep both: the best solution and the average of the best solutions.
While (error > $\varepsilon$ and the number of iteration is not meet) do
Begin
    For each s in $S_1$ do
    Begin
        Compute a new solution by applying case 1 and 2 using Equations (2) and (3).
        If (the new solution is better than the current solutin (s))
          Then, s = new solution.
    End
    For each s in $S_2$ do
    Begin
        Compute a new solution by applying case 3 and 4 using Equations (4) and (5).
        If (the new solution is better than the current solutin (s))
          Then, s = new solution.
    End
    For each s in $S_3$ do
    Begin
        Compute a new solution by applying case 5 using Equations (6) and (7).
        If (the new solution is better than the current solutin (s))
          Then, s = new solution.
    End
    For each s in $S_4$ do
    Begin
        Generate a new random solution by applying case 6 using Equations (8).
        If (the new solution is better than the current solutin (s))
          Then, s = new solution.
    End
    If (any produced solution in ($S_1$, $S_2$, $S_3$, or $S_4$) is superior to the best solution)
          Then, best solution = new solution.
    Update the average of the best solution.
End
Return the best solution.
```

Figure 2:
The main steps of the CFA

3. THE PROPOSED CFA FOR RULE GENERATOR

This paper aims to use the CFA as a rule generator for IDS problem. The generated rules are then used to classify the instances to one of the five class labels in the KDD-Cup-99 dataset: Normal, Dos, Probing, R2L, and U2R (Tavallaee, 2009). First, the training dataset is divided into five groups according to the number of class labels in the KDD-Cup-99. Then, for each feature in each group, the maximum and the minimum values are calculated to produce the two vectors $Max_c[N]$ and $Min_c[N]$, where $c = 1, 2, \ldots, C$ and C is the number of classes, and N presents the number of features in each sample. In the training stage, each newly generated rule (Upper and Lower) for each group of training data at each step of the CFA including the initialization process, the newly generated rule is tested using the training data set as follows. If any sample belonging to the group (i) is satisfied by the newly generated rule, then remove this sample from the group (i) and recalculate the Max and Min vectors. This process is repeated for all groups until all samples are removed. The initialization process and the work of the CFA are described in the following sections.

3.1 Initialization

The population $P[M]$ is initialized with M solutions, where $P = \{p_1, p_2, \ldots, p_M)$ and M is the size of the population P. Each p_i contains a class name and two vectors, *Upper[N]* and *Lower[N]*, and N is the number of features in each instance. *Upper[N]* presents the upper boundary of this rule, while *Lower[N]* presents the lower boundary. The initialization process of *Upper* and *Lower* vectors for each p_i are calculated using (9), (10) and (11) as follows:

$$mid[n] = (Max[n] + Min[n])/2 \quad (9)$$
$$Upper[n] = random * (Max[n] - mid[n]) + mid[n] \quad (10)$$
$$Lower[n] = random * (mid[n] - Min[n]) + Min[n] \qquad n = 1,2,\cdots,N \quad (11)$$

where $Max[n]$ is the maximum, and $Min[n]$ is the minimum values of feature n, *random* is a random number to be generated between (0, 1). In the original CFA, the population is divided into four subsets; however, in this study, the population is divided into three subsets S_1, S_2, and S_3, because case 5 is used for the pruning rule, thus we only need three subsets in our modified CFA. After the initialization step, the processes of the six cases of the CFA are applied as follows.

3.2 Application of case 1 and 2 on S_1

In cases 1 and 2, the light reflection process occurs because of the association between the chromatophores layer and the iridophores layer, and they are used for the global search. In the original CFA, R simulates the stretching process of saccule while V simulates the last view of the matched pattern. In order to use the CFA as a rule generator for the IDS classification problem, (2) and (3) which have been used to find the reflection and the visibility in each element in S_1 are modified as given below:

$$ref_1 = R * record[n] \quad (12)$$
$$vis_1 = V * (Max[n] - Min[n]) \quad (13)$$
$$ref_2 = R * record[n] \quad (14)$$
$$vis_2 = -V * (Max[n] - Min[n]) \quad (15)$$

where *Max[n]* and *Min[n]* are defined for feature n, *record[n]* is any random value of feature n selected randomly from the training dataset. R is 1 and V is generated randomly between the interval (0, 1), with 0.2 probability *newUpper* and *newLower* values of new solutions are calculated using (1) as follows:

$$S_1[i].newUpper[n] = ref_1 + vis_1 \quad (16)$$
$$S_1\ [i].newLower[n] = ref_2 + vis_2 \quad (17)$$

where $i = 1, 2, \ldots, S_1.$ *Size.*

While, with the probability of 0.8, the other cases are used to produce the new solution. Sometimes, the value of the newly generated rule (*newUpper*$[n]$ or *newLower*$[n]$) is out of range. In this case, any selected random value from *feature*$[n]$ can be satisfied.

3.3 Application of case 3 and 4 on S_2

The iridophores are the reflective cells. They reflect light for the concealment the organs, which means that the outgoing light must be close to the environment. Therefore, the incoming light is displayed as a feature value and is revised with a small difference. The simulation of this process is reformulated in (18)-(21) as follows:

$$ref\_1 = R * record[n] \quad (18)$$
$$vis\_1 = V * (Max[n] - record[n]) \quad (19)$$
$$ref\_2 = R * record[n] \quad (20)$$
$$vis_2 = -V * (record[n] - Min[n]) \quad (21)$$

where *Max*$[n]$ and *Min*$[n]$ are the respectively the max and the min values of feature n, *record* is an instance that is selected randomly from the training dataset. The R value is equal to 1, but the V is generated randomly from the interval (0, 1). Then the new *newUpper* and *newLower* are calculated using (1) as follows:

$$S_2\ [i].newUpper[n] = ref_1 + vis_1 \quad (22)$$
$$S_2\ [i].newLower[n] = ref_2 + vis_2 \quad (23)$$

where $i = 1, 2, \ldots, S_2$.

3.4 Application of case 6 on S_3

In the original CFA, case 5 is used before case 6. In this study, case 5 is used for the rule pruning process, which is described in section 3.5. CFA uses case 6 as the global search; hence any random solution is satisfactory. In this study, the same Equations (9), (10), and (11) that are used in the initialization process are reused here for S_3.

3.5 Application of case 5 for rule pruning

Rule pruning is an important task to increase the accuracy of the model and enhancing the quality of the produced rule itself. It can be used to remove irrelevant information from the rule. The objective of rule pruning is to evacuate redundant or unnecessary features from the dataset, which may negatively affect the results and the performance of the model. The study of (Eesa, 2015) is based on using CFA for feature selection. The authors have successfully used case 5 of CFA to remove one feature at a time and evaluate the remaining features. If the remaining features produce some better results, they are kept, and another feature is removed. This procedure is repeated until all features are examined; hence the most relevant features are selected. In this study, we have used the same method. At each time a sub-rule is removed from the current rules, then the remaining sub-rules will be tested using the training dataset. If the remaining rules are produced a better result, then another sub-rule is removed and so on until all sub-rules are examined. For more description, consider a vector called *Flag* with the size of *N* is selected, and *N* represents all features considered in the training dataset, and let *x* to be a rule that belongs to class *c*. At each time the rule of feature *n* which is represented by *x.Upper[n]* and *x.Lower[n]* is removed from *x.Upper* and *x.Lower*. If *x* produces some better results, *Flag[n]* is set to be 0; otherwise, *Flag[n]* is 1. This indicates that any feature with *Flag[n]*=1, their values of *x.Upper[n]* and *x.Lower[n]* are considered. To assess the quality of the produced rules, the following fitness method is used:

$$Fitness = TP/(TP + FN) * TN/(TN + FP) \tag{24}$$

where *TP* and *TN* indicate the quantity of true positive and true negative instances which are classified effectively, whereas *FP* and *FN* indicate the number of instances that are incorrectly classified as a false positive and false negative, respectively (Khraisat, 2019; Sumaiya Thaseen & Aswani Kumar, 2017). The classification process will be described next in Section 3.6. The general steps of the rule pruning process are described in the procedure shown in Figure 3.

3.6 Classification using the generated rules

After applying the rule pruning process and removing the unnecessary sub-rules, the pruned rules are used to classify each instance in the testing data to one of the five class labels in the KDD-Cup-99-dataset: Normal, Dos, Probing, U2R, and R2L. The classification process works as follows: If all features' values of record *r* are covered by the rule *x* of class *c* so that all values are between the *x.Upper* and *x.Lower*, then *r* is classified as class *c*. However, this is not always the case, as sometimes one instance in the testing data may be involved by more than one rule for various classes. In such a case, the *bias-value* is calculated for all the covered rules. Then these values are accumulated according to different possible classes. The class with the greatest *bias-value* is chosen to be the true predicted class. The calculation of *bias-value* is formulated in (25).

```
For each class $c$ in $C$ do
Begin
    For each rule $x$ belonging to class $c$ do
    Begin
        For each Upper and Lower of feature $n$ do
        Begin
            Remove $x.Upper[n]$ from $x.Upper$
            Remove $x.Lower[n]$from $x.Lower$
            Evaluate $x$ using the Fitness function
            IF $x$ is produced better result Then
                $x.Flag[n]$ =0
            Else
                $x.Flag[n]$ = 1
        End
    End
End
```

Figure 3:
Rule pruning process using case 5.

$$bias\_value = a * Rule.fitness + b * Rule.accuracy \qquad (25)$$

where *a* and *b* are two weighted values to be determined by the user. The value of *a* is between (0, 1), while the value of *b* is equal to (1-*a*). In this study, the values of *a* and *b* are set to 0.5. The value of the *fitness* metric is calculated using (24), and for each covering rule belonging to class *c*, accuracy is calculated using equation (26),

$$accuracy = \frac{No.of\ correctly\ classified\ record\ as\ class(c)}{No.of\ all\ records\ that\ belonging\ to\ class\ (c)} \qquad (26)$$

4. EXPERIMENTAL SETUP

To assess the efficiency of the proposed method, we have compared its performance with four traditional classifiers: DT, SVM, K-NN, and NB. The proposed method is experimentally assessed using the KDD-Cup-1999 dataset, which is obtained from the UCI machine learning repository (UCI Machine Learning Repository, 2015). The experiments are performed in order to present that our proposed method is generally feasible to be used for rule generation and hence it can be used as a new classifier for IDS.

4.1 Data preparation

The “10%KDD-Cup-99” is a very popular dataset commonly used for benchmarking intrusion detection problems (Tavallaee et al., 2009). The dataset contains 494,020 training and 311,028 testing connection records (Eesa, 2015). Each record contains 41 independent features, and it is labelled as one of the five classes considered in the “KDD-Cup-99” dataset: Normal, Dos, Probing, R2L and U2R. Where Dos (denial of service) is a type of attack that causes some computing or memory resource to be busy or too full to handle legitimate requests. Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. R2L (Remote to Local) is a class of attacks where an attacker sends packets to a machine over a network, then exploits the machine vulnerability to illegally gain local access as a user. While with the U2R (User to Root) attack, a normal account is used by an attacker to login into the system of a victim and tries to gain administrator privileges by exploiting someone vulnerability in the victim.

Table 1. Different types of attacks and their corresponding occurrence number, respectively in the training and testing subsets chosen from KDD Cup 99 dataset

Normal (937; 606)	
Probing (41; 42)	psweep(12;3), Mscan(0;11), Nmap(2;1), Portsweep(11;4), Saint(0;7), Satan(16;16).
DoS(3915 ; 2299)	apache2(0;8), back(22;11), land(0; 0), mailbomb(0;50), Neptune(1072;580), processtable(0;8), Pod(3;1), udpstorm(0;0), Smurf(2808;1641), Teardrop(10;0),
U2R(5 ; 10)	buffer_overflow(3;1), httptunnel(0;3), loadmodule(0;0), perl(0;0), rootkit(2;2), xterm(0;2), Ps(0;2), Sqlattack(0;0),
R2L(13; 160)	ftp_write(0;0), imap(0;0), guesspasswd(2;44), named(0;0), multihop(0;0), phf(0;0), sendmail(0;0), snmpgetattack(0;77), snmpguess(0;24), spy(0;0,), warezclient(10;0), worm(0;0), warezmaster(1;15), xsnoop(0;0), xlock(0;0),

However, this dataset is too big to be used in such experiments. Therefore, the training and the testing data are chosen randomly from the 10%KDD-Cup-99 dataset to be utilized for our experiment. Table 1 describes the amount of each attack class in the chosen training and testing subsets. In order to keep the same proportion of data, each attack is divided by 100 (Eesa, 2015) in the training and testing datasets. In Table 1, Psweep (12, 3) means that this attack has 12 attacks in the training set and 3 attacks in the testing set. In our study, all categorical values in the datasets are converted to numerical values. For example, the protocol_type attribute consists of three categorical values (tcp, udp, icmp), and these values are converted to (10, 20, and 30), respectively. For instance, if an attribute consists of 100 categorical values, these values are converted to (10, 20, 30, …, 1000), respectively.

4.2 Evaluation

In order to assess the effectiveness of the proposed method using CFA classification model, five well-known metrics are used in our evaluation process; namely "True Positive Rate" (TPR), "False Positive Rate" (FPR), Precision, Recall, and Area Under the Curve (AUC) (Jiao & Du, 2016). Then, the efficiency of the proposed CFA method is compared with four well-known techniques in Weka (Hall et al., 2009), namely DT, K-NN, SVM, and NB. The formulas of the five-evaluation metrics are stated below:

$$TPR_i = TP_i \;/\; (TP_i + FN_i)$$
$$FPR_i = FP_i \;/\; (FP_i + TN_i)$$
$$Precision_i = TP_i \;/\; (TP_i + FP_i)$$
$$Recall_i = TP_i \;/\; (TP_i + FN_i)$$
$$AUC_i = (1 + TPR_i - FPR_i) \;/\; 2$$

where i = 1, 2, …, C, and C is the number of classes.

5. EXPERIMENTAL RESULTS

The proposed method is implemented using C# language within the Microsoft Visual Studio environment. The population size is set to 10. First, the validation of the proposed model is tested for 10 independent runs. Table 2 describes the obtained results in terms of TPR metric for each run. It can be noticed that our proposed CFA classification method has successfully classified the KDD-Cup-99 data, and it obtains a good result where TPR is varied between 91.24 and 92.71, and the average overall 10 independent runs is equal to 92.203.

Table 2. Experimental results for 10 independent runs using the proposed method

Runs	TPR
Run#1	92.651
Run#2	91.239
Run#3	92.581
Run#4	91.302
Run#5	92.489
Run#6	92.460
Run#7	92.548
Run#8	92.591
Run#9	92.711
Run#10	91.462
Best	**92.711**
Worst	**91.239**
Average	**92.203**

Table 3 illustrates the comparison results of the proposed method with the other four techniques DT, K-NN, SVM, and NB. The comparison results based on the for metrics (FPR, Precision, Recall, AUC) are detailed in Table 3 and graphically shown in Figures 4 and 5. All the reported performance results of the proposed method in Table 3 are averaged over 10 independent runs. From Table 3 and Figures 4 and 5, it can be seen clearly that the new CFA method has provided better results than all other classification methods in terms of Precision, Recall, and AUC with a lower FPR. Since the AUC metric is commonly used to distinguish the performance of more than one model (Tharwat, 2018), we can conclude that our proposed method was successful in terms of this evaluation metric, as illustrated obviously in Figure 5.

Table 3. The comparison of classification results for the proposed method with different methods

Techniques	TPR	FPR	Precision	Recall	AUC
DT	0.918	0.028	0.931	0.918	0.936
SVM	0.746	0.024	0.911	0.746	0.790
K-NN	0.740	**0.022**	0.878	0.740	0.874
NB	0.703	0.186	0.855	0.703	0.803
Proposed method	**0.922**	0.027	**0.939**	**0.922**	**0.947**

Besides, in order to further investigate the efficiency and performance of the newly proposed CFA method, we compared the obtained results with our previews work (Eesa, 2015). Table 4 illustrates the comparative results in terms of TPR evaluation metric.

Table 4. The comparison of the proposed method with (Eesa, 2015)

Method	Number of features	TPR
method in (Eesa, 2015)	41	71.087
	35	69.526
	30	69.538
	25	78.212
	20	91.362
	15	91.500
	10	92.051
	5	91.000
Proposed method	41	**92.203**

Results in Table 4 present that the newly proposed method has performed better than our previous study [12] in terms of TPR, even when using different numbers of features. For instance, although the previous method has provided the highest TPR of 92.051, our new method can provide higher TPR than that without using feature selection. These results suggest that even without using any feature selection technique, the newly proposed method performs better.

From the obtained results, we believe that CFA can be used as an alternative tool for data mining in the IDS domain. The proposed CFA method has been tested many times in many different experiments, and it has provided the same results with (±0.2) errors which present the robustness and the stability of the proposed model.

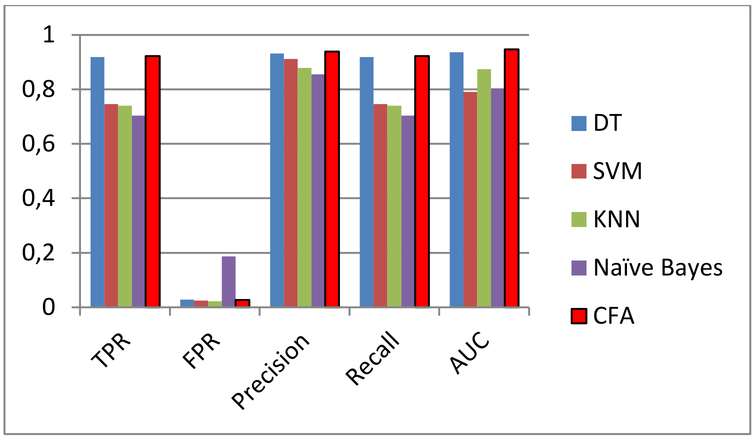

Figure 4:
Bar chart of different methods with different evaluation metrics

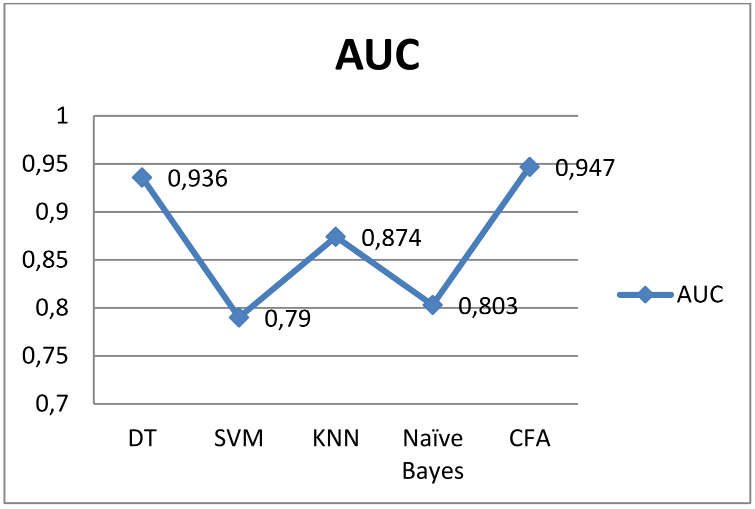

Figure 5:
The AUC results for all different methods

6. CONCLUSION AND FUTURE WORK

In this paper, we investigated the use of the modified CFA for IDS as a rule generation tool. The CFA was modified to generate a set of rules for each class considered in the dataset. One of the fundamental features of the CFA method is its simplicity as the generated rules are only represented by two vectors *Upper* and *Lower*, and they can easily be used for the classification task. In order to check the efficiency of the proposed method, we used the "KDD-Cup-99" dataset. The obtained results were promising and showed the robustness and effectiveness of the proposed method. The achieved results were assessed utilizing the five-performance metrics TPR, FPR, Precision, Recall, and AUC. Experimental results also demonstrated that the new CFA offers a very competitive method in comparison with many traditional classification methods. During the experiments, we observed that the proposed method was time-consuming to find rules. The execution time for training and testing processes for each run took about 20 seconds while the running times for DT, SVM, K-NN and NB were 0.3, 1, 6, 0.4 seconds, respectively. This limitation can be considered as future work to be further investigated. In addition, the proposed CFA data mining method can be utilized to address the classification problems in different domains. Based on the above analysis, excluding the execution time, we conclude that CFA is a considerable potential rule-generation tool.

CONFLICT OF INTEREST

The authors acknowledge that there is no conflict of interest or common interest with any institution/organization or person.

AUTHOR CONTRIBUTION

Adel Sabry Eesa contributed to the determination and management of the conceptual and design processes of the study, data collection, data analysis and interpretation, creation of the draft paper, critical analysis of the intellectual content and final approval with full responsibility.

Sheren Sadiq, contributed to the determination and management of the conceptual and design processes of the study, data collection, data analysis and interpretation, creation of the draft paper, critical analysis of the intellectual content and final approval with full responsibility.

Masoud Muhammed contributed to the determination and management of the conceptual and design processes of the study, data collection, data analysis and interpretation, creation of the draft paper, critical analysis of the intellectual content and final approval with full responsibility.

Zeynep Orman contributed to the determination and management of the conceptual and design processes of the study, data analysis and interpretation, creation of the draft paper, critical analysis of the intellectual content and final approval with full responsibility.

REFERENCES

1. Aburomman, A.A. and Reaz, M.B.I. (2016) A novel SVM-kNN-PSO ensemble method for intrusion detection system, *Applied Soft Computing Journal*, 38, 360–372. doi:10.1016/j.asoc.2015.10.011

2. Aghdam, M. H. and Kabiri, P. (2016) Feature Selection for Intrusion Detection System Using Ant Colony Optimization, *International Journal of Network Security,* 18(3), 420-432. https://pdfs.semanticscholar.org/022d/50ecb37eb6c78be9728ed7bc198a29cc6915.pdf

3. Ali, G.A. and Jantan, A. (2011) A New Approach Based on Honeybee to Improve Intrusion Detection System Using Neural Network and Bees Algorithm, *International Conference on Software Engineering and Computer Systems,* Springer, Berlin, Heidelberg, 777–792. doi:10.1007/978-3-642-22203-0_65

4. Arshak, Y., and Eesa, A. (2018) A New Dimensional Reduction Based on Cuttlefish Algorithm for Human Cancer Gene Expression, *International Conference on Advanced Science and Engineering*, IEEE, Duhok, Iraq, 48-53. doi: 10.1109/ICOASE.2018.8548908

5. Balasaraswathi, V.R., Sugumaran, M. and Hamid, Y. (2018) Chaotic Cuttle Fish Algorithm for Feature Selection of Intrusion Detection System. *International Journal of Pure and Applied Mathematics*, 119(10), 921–935. https://acadpubl.eu/jsi/2018-119-10/articles/10a/81.pdf

6. Chung, Y.Y. and Wahid, N. (2012) A hybrid network intrusion detection system using simplified swarm optimization (SSO), *Applied Soft Computing*, 12(9), 3014–3022. doi:10.1016/J.ASOC.2012.04.020

7. Duric, Z. (2014) WAPTT - Web Application Penetration Testing Tool, *Advances in Electrical and Computer Engineering*, 14(1), 93–102. doi:10.4316/AECE.2014.01015

8. Eesa, A.S., Abdulazeez, A.M.A., and Orman, Z. (2017) A DIDS Based on The Combination of Cuttlefish Algorithm and Decision Tree, *Science Journal of University of Zakho*.

doi:10.25271/2017.5.4.382

9. Eesa, A.S., Brifcani, A.M.A and Orman, Z. (2014) A New Tool for Global Optimization Problems-Cuttlefish Algorithm, *International Journal of Computer and Information Engineering, World Academy of Science, Engineering and Technology*, *8*(9), 1235–1239. https://waset.org/publications/9999515/a-new-tool-for-global-optimization-problems-cuttlefish-algorithm

10. Eesa, A.S. and Orman, Z. (2020), A new clustering method based on the bio-inspired cuttlefish optimization algorithm, *Expert Systems*, 37, 1-13. doi:10.1111/exsy.12478

11. Eesa, A.S., Orman, Z. and Brifcani, A.M.A. (2015) A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems, *Expert Systems with Applications*, 42(5), 2670–2679. doi:10.1016/J.ESWA.2014.11.009

12. Gauthama, R.M.R., Somu, N., Kirthivasan, K., Liscano, R. and Shankar S.V.S. (2017) An efficient intrusion detection system based on hypergraph - Genetic algorithm for parameter optimization and feature selection in support vector machine, *Knowledge-Based Systems*, 134, 1–12. doi:10.1016/j.knosys.2017.07.005

13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10. doi:10.1145/1656274.1656278

14. Hamamoto, A.H., Carvalho, L.F., Sampaio, L.D.H., Abrão, T. and Proença, M.L. (2018) Network Anomaly Detection System using Genetic Algorithm and Fuzzy Logic, *Expert Systems with Applications*, 92, 390–402. doi:10.1016/J.ESWA.2017.09.013

15. Issa, A.S. and Brifcani, A.M. (2011) Intrusion Detection and Attack Classifier Based on Three Techniques: A Comparative Study, *Engineering and Technology Journal*, 29(2), 386–412. https://www.iasj.net/iasj?func=article&aId=26174

16. Jiao, Y. and Du, P. (2016) Performance measures in evaluating machine learning based bioinformatics predictors for classifications, *Quantitative Biology*, 4(4), 320–330. doi:10.1007/s40484-016-0081-2

17. Jose, S., Malathi, D., Reddy, B. and Jayaseeli, D. (2018) A Survey on Anomaly Based Host Intrusion Detection System, *Journal of Physics: Conference Series*, 1000(1), 012049. doi:10.1088/1742-6596/1000/1/012049

18. Kanaka, V.K. and Sitamahalakshmi, T. (2017) Implementation of Intrusion Detection System Using Artificial Bee Colony with Correlation-Based Feature Selection, *Advances in Intelligent Systems and Computing*, Springer, Singapor, 507, 107–115. doi:10.1007/978-981-10-2471-9_11

19. Khraisat, A., Gondal, I. and Vamplew, P. (2018) An Anomaly Intrusion Detection System Using C5 Decision Tree Classifier, Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, Cham, 149–155. doi:10.1007/978-3-030-04503-6_14

20. Khraisat, A., Gondal, I., Vamplew, P. and Kamruzzaman, J. (2019) Survey of intrusion detection systems: techniques, datasets and challenges, *Cybersecurity*, 2(1), 20. doi:10.1186/s42400-019-0038-7

21. Kiziloluk, S. and Alatas, B. (2015) Automatic mining of numerical classification rules with parliamentary optimization algorithm, *Advances in Electrical and Computer Engineering*, 15(4), 17–24. doi:10.4316/AECE.2015.04003

22. Koc, L., Mazzuchi, T.A. and Sarkani, S. (2012) A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier, *Expert Systems with Applications*, 39(18), 13492–13500. doi:10.1016/J.ESWA.2012.07.009

23. Li, W., Yi, P., Wu, Y., Pan, L. and Li, J. (2014) A new intrusion detection system based on KNN classification algorithm in wireless sensor network, *Journal of Electrical and Computer Engineering, 2014*.doi:10.1155/2014/240217

24. Li, Yang and Guo, L. (2007) An active learning based TCM-KNN algorithm for supervised network intrusion detection, *Computers and Security*, 26(7–8), 459–467. doi:10.1016/j.cose.2007.10.002

25. Li, Yinhui, Xia, J., Zhang, S., Yan, J., Ai, X. and Dai, K. (2012) An efficient intrusion detection system based on support vector machines and gradually feature removal method, *Expert Systems with Applications*, 39(1), 424–430. doi:10.1016/j.eswa.2011.07.032

26. Mukherjee, S. and Sharma, N. (2012) Intrusion Detection using Naive Bayes Classifier with Feature Reduction, *Procedia Technology*, 4, 119–128. doi:10.1016/J.PROTCY.2012.05.017

27. Panigrahi, R. and Borah, S. (2018) Rank Allocation to J48 Group of Decision Tree Classifiers using Binary and Multiclass Intrusion Detection Datasets, *Procedia Computer Science*, 132, 323–332. doi:10.1016/j.procs.2018.05.186

28. Patel, K. and Buddhadev, B. (2015) Predictive rule discovery for network intrusion detection, *Advances in Intelligent Systems and Computing*, 321, 287–298. doi:10.1007/978-3-319-11227-5_25

29. Eesa, A.S., Brifcani, A.M.A and Orman, Z. (2013) Cuttlefish Algorithm – A Novel Bio-Inspired Optimization Algorithm, *International Journal of Scientific & Engineering Research*, 4(9), 1978-1986. https://www.ijser.org/onlineResearchPaperViewer.aspx?Cuttlefish-Algorithm-A-Novel-Bio-Inspired-Optimization-Algorithm.pdf

30. Schuh, G., Reinhart, G., Prote, J.P., Sauermann, F., Horsthofer, J., Oppolzer, F. and Knoll, D. (2019) Data mining definitions and applications for the management of production complexity, *Procedia CIRP*, 81, 874–879. doi:10.1016/j.procir.2019.03.217

31. Sumaiya, T.I. and Aswani, K.C. (2017) Intrusion detection model using fusion of chi-square feature selection and multi class SVM, *Journal of King Saud University - Computer and Information Sciences*, 29(4), 462–472. doi:10.1016/J.JKSUCI.2015.12.004

32. Swarnkar, M. and Hubballi, N. (2016) OCPAD: One class Naive Bayes classifier for payload based anomaly detection, *Expert Systems with Applications*, 64, 330–339. doi:10.1016/j.eswa.2016.07.036

33. Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A. (2009) A detailed analysis of the KDD CUP 99 data set, *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6. doi:10.1109/CISDA.2009.5356528

34. Tharwat, A. (2018) Classification assessment methods, *Applied Computing and Informatics*. https://doi.org/10.1016/j.aci.2018.08.003

35. UCI Machine Learning Repository (2015) *KDD Cup 1999 Data*. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

36. Vancea, F. (2014) Intrusion Detection in NEAR System by Anti-denoising Traffic Data Series using Discrete Wavelet Transform, *Advances in Electrical and Computer Engineering*, 14(4), 43–48. doi:10.4316/AECE.2014.04007

37. Varma, P.R.K., Kumari, V.V. and Kumar, S.S. (2016) Feature Selection Using Relative Fuzzy Entropy and Ant Colony Optimization Applied to Real-time Intrusion Detection

System, *Procedia Computer Science,* 85, 503–510. doi:10.1016/J.PROCS.2016.05.203

38. Zhang, J., Ling, Y., Fu, X., Yang, X., Xiong, G. and Zhang, R. (2020) Model of the intrusion detection system based on the integration of spatial-temporal features, *Computers and Security*, *89*, 101681. doi:10.1016/j.cose.2019.101681

39. Zhao, M., Zhai, J. and He, Z. (2010) Intrusion detection system based on support vector machine active learning and data fusion, *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 272–279. doi:10.1007/978-3-642-16493-4_28