# Application of Fixed Structure Learning Automata For Designing Intrusion Detection Systems

Kayvan Asghari*

## Abstract

Designing an efficient intrusion detection system includes several important phases, feature selection being one of the most important ones. In this paper, a fixed structure learning automata has been applied for the feature selection phase. The introduced method does the exploration and exploitation phases of an optimization method perfectly, finding the most important features of the network events to detect intrusions. The count of selected features in the proposed method is a pre-defined number as the feature selection is a multi-objective problem, and one of the important objectives is the feature count. The learning automata-based method uses reward and penalty operators to explore the problem's search space. The proposed method tries to enhance the accuracy rate for intrusion detection, which is another significant objective for a feature selection method. A well-known intrusion detection dataset called the NSL-KDD has been used in this paper to evaluate the proposed method. The evaluation results indicate the acceptable performance of the proposed method compared with some of the existing ones.

**Keywords:** Fixed structure learning automata, Feature selection problem, Intrusion detection system.

## 1 Introduction

Various optimization algorithms have been proposed in recent years to solve different optimization problems. These algorithms are not general approaches for all optimization problems despite having common characteristics. Each of these algorithms has weaknesses that prevent them from reaching optimal solutions. One of the main problems of these algorithms is the failure to improve the solutions in different iterations and the problem of getting stuck in the local optimum. The general operational pattern of optimization algorithms is the discovery of solutions in the early iterations of the algorithm by making diversity for a complete scan of the problem search space. However, in the last iterations, these algorithms change their strategy and search around the solutions found in the previous steps to reach the global optimal solution. The mentioned phases in the algorithm in various literature are called exploration and exploitation. The time to change between the exploration and exploitation phases and how to manage this issue is significant for an optimization algorithm, which makes the algorithm work better. Controlling the balance between exploration and exploitation is performed by some variable parameters of the algorithm, which are changed during the learning process. These parameters force the optimization algorithm to explore the problem's search space in the first iterations. But, in the ending iterations and during the exploitation, they compel the algorithm to perform a local search around the best-found solutions in the exploration phase. In ideal conditions, the algorithm should generate diverse solutions with long moves in the search space at the beginning and gradually converge to near-optimal solutions in the final iterations. The growing use of network applications and web services has increased the risk of intrusion attacks in different networks. Thus, designing accurate and reliable intrusion detection systems is a significant task to have secure networks. Various types of intrusion detection systems have been introduced so far. One of the well-known categorizations is the anomaly-based and signature-based systems, while the second category is investigated in this paper. Selecting the important features that must be checked by the intrusion detection system is an important phase of its design. These features are used to build an event classifier that must be minimal, quick, and accurate. Various feature selection methods have been proposed in the literature so far [1–7]. The application of fixed structure learning automata for solving the feature selection problem of designing an intrusion detection system has been presented in this paper. The naive Bayesian network has been employed to evaluate the selected features of the compared algorithms. The environment is the evaluation function of selected features using the naive Bayesian network for the implemented learning automata-based algorithm. During the iterations of the learning process, the automata interact with the environment and improve their actions by employing the penalty and reward mechanisms.

*Department of Computer Engineering, Sardroud Center, Tabriz Branch, Islamic Azad University, Tabriz, Iran, k.asghari@iau.ac.ir, k.asghari@yahoo.com

## 2 Related works

A brief explanation of the learning automata tool, including its types and the feature selection problem in developing intrusion detection systems, is presented in this section.

### 2.1 The learning automaton

A learning automaton is a conceptual learning tool, that can be implemented in different ways and used for various applications. During the learning process, it interacts with a random environment to identify the environment's internal specifications. The learning automaton tries to find out the probabilistic relationship between its actions and the environment's feedback. Thus, it selects different actions to find the optimal solution with the guidance of the environment.

There are different classifications for the learning automaton. One of them is classifying it into fixed and variable structures. The probability of changing actions and transitions between the automaton states is a fixed value in the fixed structure learning automaton. But, in the variable structure, the mentioned probabilities are updated considering the environment feedback [8]. The object migration learning automaton as a fixed structure type is used in this paper to solve the feature selection problem in intrusion detection systems [9].

The learning automaton is defined formally by the quintuple of $\{\phi, \alpha, \beta, F(\cdot,\cdot), H(\cdot,\cdot)\}$, where $\phi$ is a set of internal states $\phi = \{\phi_1, \phi_2, ..., \phi_s\}$, $\alpha$ is a set of automaton actions $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$, and $\beta$ is a set of environment responses $\beta = \{\beta_1, \beta_2, \ldots, \beta_m\}$. The finite-type state-output automata have been employed in this paper to select features in intrusion detection systems because the $\phi$, $\alpha$, and $\beta$ sets are finite.

The state, action, and response of the environment for the learning automata in the moment of $t$ is represented by $\phi(t)$, $\alpha(t)$, and $\beta(t)$, respectively. The environment responses set can be finite or infinite, which is $\beta = \{0, 1\}$ for the proposed method. '0' means an unfavorable response or penalty, while '1' means a favorable response or reward from the environment.

$F(\cdot,\cdot)$ is a state transfer function, which gets the current state and environment response and returns the next state ($F(\cdot,\cdot) : \phi \times \beta \to \phi$). Finally, $H(\cdot,\cdot)$ is a function for determining the current action with the current environment response and current state ($H(\cdot,\cdot) : \phi \times \beta \to \alpha$)) [8]. Each learning automaton in the learning automata has a finite set of states. It selects and returns an action in each iteration considering its current state. The interaction of the learning automaton with the working environment is illustrated in Figure 1.

The automaton selects a feature as its action ($\alpha_i$), where it is the input of the environment. For every action, the environment returns feedback ($\beta_i$). The feedback from the environment in moment $n$, which can be a reward ($\beta(n) = 1$) or penalty($\beta(n) = 0$), updates the state of the current action ($\phi_i$). For a reward from the environment, the probability of selected action ($\alpha_i$) increases by the learning automaton. Instead, for a penalty response, the selected action's probability is decreased [8].

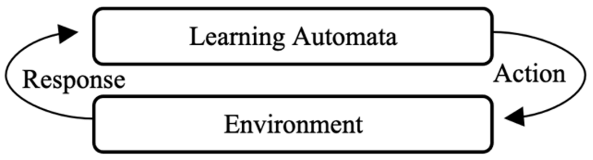

Figure 1: Interaction between the learning automata and the environment

Learning automata have various applications in different fields like pattern recognition, neural and Bayesian network optimization [10], network routing [11], job scheduling [12] query optimization [13], data compression, and solving NP problems.

## 3 The fixed structure learning automata for feature selection

The intrusion detection systems must spend a lot of energy and time classifying a network packet as an intrusion or normal network event. The reason is the establishment of each packet from tens of features, some of which are duplicated or useless. Therefore, in recent decades various feature selection methods have been introduced by researchers for designing efficient and rapid intrusion detection systems [3].

A fixed structure type of the learning automata tool is proposed in this section to find a near-optimal solution for the feature selection problem in designing intrusion detection systems. The proposed method uses the penalty and reward mechanisms to obtain a selected feature set, which will be employed to build an intrusion classifier. The selected feature set delegates all features in the network packet, which is minimal and has no duplicate information. The target classifier constructed by the selected features can inspect the input network packet to identify the intrusions from normal network events.

A naive Bayesian network [14] is employed as a simple and easy-to-implement classifier to evaluate the selected features of the fixed structure learning automata-based algorithm and other existing methods. Other kinds of classifiers like artificial neural networks can also be used for the solution evaluation, but the concentration of this paper is over the feature selection phase. To demonstrate the working procedure of the proposed method, the flowchart of Figure 2 can be useful.

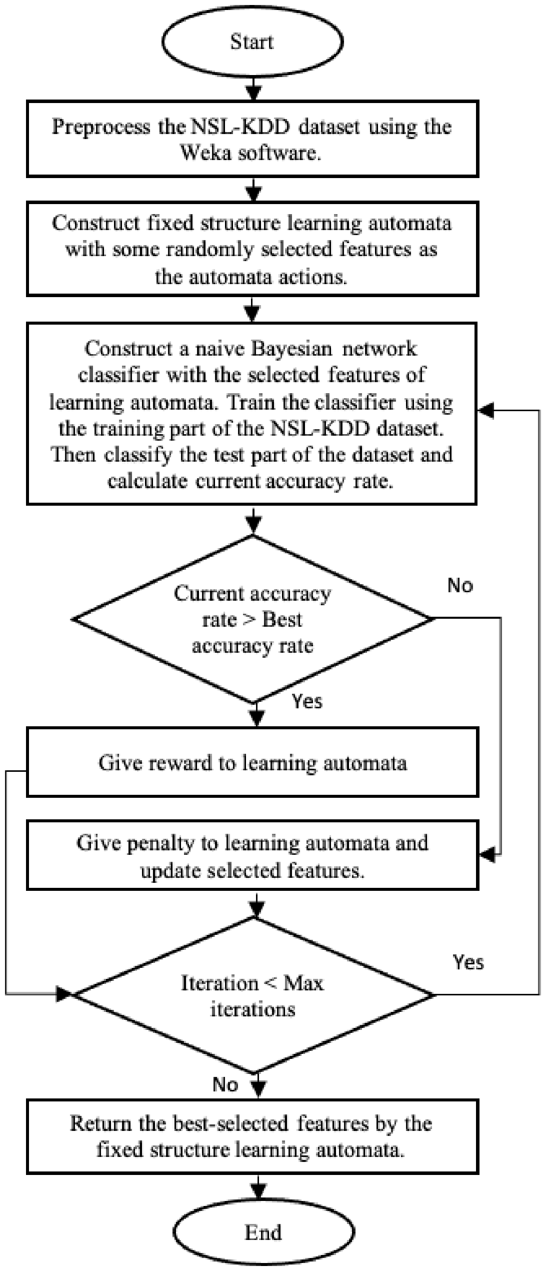
Start
Preprocess the NSL-KDD dataset using the Weka software.
Construct fixed structure learning automata with some randomly selected features as the automata actions.
Construct a naive Bayesian network classifier with the selected features of learning automata. Train the classifier using the training part of the NSL-KDD dataset. Then classify the test part of the dataset and calculate current accuracy rate.
Current accuracy rate > Best accuracy rate
No
Yes
Give reward to learning automata
Give penalty to learning automata and update selected features.
Iteration < Max iterations
Yes
No
Return the best-selected features by the fixed structure learning automata.
End

Figure 2: The fixed structure learning automata-based model for the feature selection problem

To apply an intrusion detection dataset, a pre-process of features to covert named fields to numbers and discretize the continuous field values is necessary, which is performed by Weka software. A demonstration of the fixed structure learning automata for selecting eight features is presented in Figure 3. For the selection of n features, n learning automata are needed, where each automaton has k states (k=5 in Figure 3). The selected features are represented in the proposed method by the actions of the automata or objects. Thus, each learning automaton is responsible for selecting one feature as its action. During the iterations of the proposed algorithm, the actions of automata may remain unchanged or be replaced, which causes the selection of another feature in the feature set. The action replacement process of each automaton is repeated to have a non-repetitive and distinct set of selected features. This method of using fixed-structure learning automata is also known as object migration learning automata [9]. A random feature is selected and dedicated as the action of each automaton at the beginning of the proposed algorithm, where the state of that action is a boundary state. The states that are numbered 5 in Figure 3 in all automata are boundary states. The penalty and reward mechanisms change the state of actions of the learning automata during the learning process. The selected features by the fixed structure learning automata are employed to construct a classifier using the naive Bayesian network. The classifier is trained with the training part of the intrusion detection dataset. Then, the trained classifier is used to classify the test part of the dataset. If the accuracy rate of the classifier is increased compared to the previous iteration, the learning automata will be rewarded by the decrease of the state numbers for the current actions. Figure 4 illustrates the rewarding mechanism for the actions of the learning automata of Figure 3. If the accuracy rate of the classifier is decreased in contrast to the previous iteration, the learning automata will be penalized by the increase of the current actions' state numbers.

Figure 5 depicts the penalizing mechanism for the actions of the learning automata of Figure 3. Features F4 and F27 in Figure 3, which were in the boundary state of the learning automata, are replaced with F6 and F25 features in Figure 5 by the penalty mechanism. The fixed structure learning automata-based algorithm includes the exploration and exploitation phases. Thus, it is a global search method. In the first iterations of the proposed algorithm, all the actions of automata or selected features are in the boundary state and may change frequently. Thus, the exploration phase is performed in the beginning. After some iterations, the state numbers of selected features with positive effects on the quality of the final solution are decreased due to the received rewards. Then, only the features on boundary states change repetitively, and the exploitation phase is performed. The proposed method balances the exploration and exploitation operations to find the near-optimal solution. The fixed structure learning automata-based algorithm explores the search space of the feature selection problem thoroughly to find promising solutions. The pseudo-code of the introduced approach for the feature selection problem of intrusion detection systems is presented in Algorithm 1.

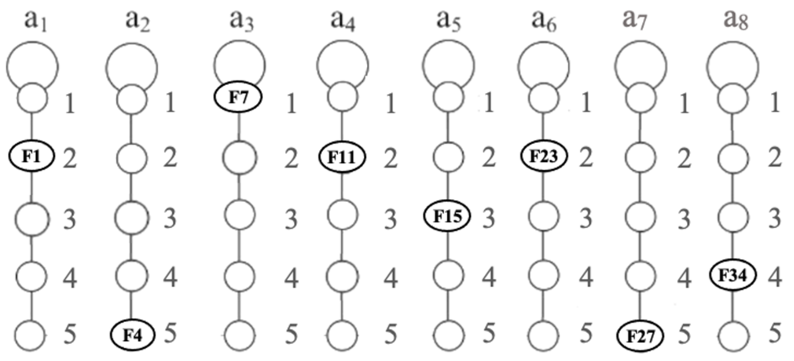

Figure 3: The fixed structure learning automata for feature selection

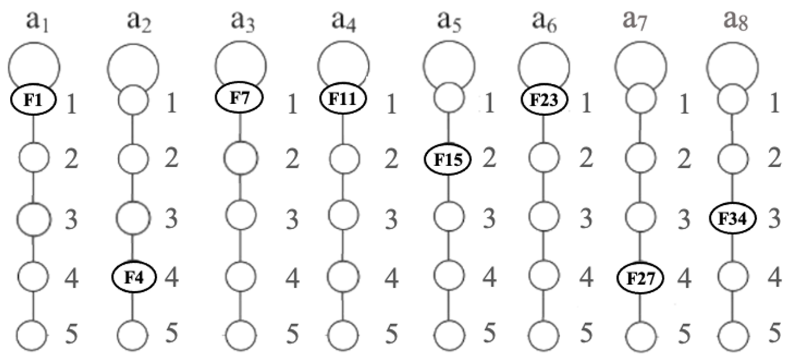

Figure 4: The reward for fixed structure learning automata for feature selection

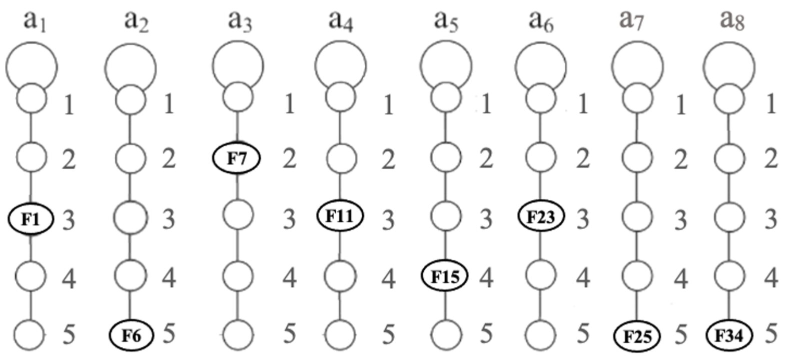

Figure 5: The penalty for fixed structure learning automata for feature selection

**Algorithm 1** Pseudo-code of the proposed fixed structure learning automata-based approach

```
Define SF[i]; (Currently Selected Features array, where $1 \leq i \leq$ Count of Selected Features)
Create $FSLA_i$ (i'th Fixed Structure Learning Automaton, where $1 \leq i \leq$ Count of Selected Features);
$AR_{Best} = 0$;
Define BestSelectedFeatures[i]; ($1 \leq i \leq$Count Of Selected Features);
$Iter = 1$;
for $i = 1$ to Count Of Selected Features do
    $FSLA_i.Action$ = Select a feature from all features set randomly, which has not been selected before;
    $SF[i] = FSLAi.Action$;
end for
while $Iter < Maximum\ Iterations$ do
    CurrentClassifier = Construct an intrusion detection naive Bayesian classifier using SF;
    Train CurrentClassifier using the training part of the intrusion detection dataset;
    $AR_{Current}$ = Classify the test part of the intrusion detection dataset with CurrentClassifier and calculate the accuracy rate;
    if $AR_{Current} > AR_{Best}$ then //(Reward)
        $AR_{Best} = AR_{Current}$;
        BestSelectedFeatures = SF;
        for $i = 1$ to Count of Selected Features do
            if $FSLA_i.CurrentState\ ! = 1$ then
                $FSLAi.CurrentState = FSLAi.CurrentState - 1$;
            end if
        end for
    else//(Penalty)
        for $i = 1$ to Count of Selected Features do
            if $FSLA_i.CurrentState$ is a boundary state then
                $FSLA_i.Action$= Select a feature randomly that is not in SF;
                $SF[i] = FSLA_i.Action$;
            else
                $FSLA_i.CurrentState = FSLA_i.CurrentState + 1$;
            end if
        end for
    end if
    Iter++;
end while
Return BestSelectedFeatures;
```

Table 1: Content of NSL-KDD dataset

| Type | Full name | Attack types | Train records | Test records |
|---|---|---|---|---|
| Normal | Normal event | – | 67343 | 9711 |
| DOS | Denial Of Service | Teardrop, Neptune Smurf | 45927 | 7458 |
| Probe | Probing attack | Satan, Saint, Portsweep | 11656 | 2421 |
| U2R | User to Root | Rootkit, Loadmodule, Buffer overflow | 995 | 533 |
| R2L | Remote to Local | Xsnoop, Password, Httptunnel | 52 | 2421 |
| Total count | | | 125973 | 22544 |

In Algorithm 1, an integer array for currently selected features called $SF$ is created at first. Then, the learning automata are constructed based on the number of selected features. The $AR_{Best}$ variable represents the best accuracy rate found so far, and the $Iter$ defines the current iteration of the algorithm. In the first $for$ loop of the algorithm, the learning automata selects the features relative to their actions. For each iteration of the algorithm in the while loop, a naive Bayesian network classifier named $CurrentClassifier$ is constructed using the selected features in the $SF$ array. The $CurrentClassifier$ is trained in the next step by the training part of the intrusion detection dataset. Then, it is evaluated by classifying the test part to calculate the accuracy rate. Next, the accuracy rate of the $CurrentClassifier$ is analogized with $AR_{Best}$. The learning automata are rewarded if the current accuracy rate exceeds the $AR_{Best}$. Otherwise, all learning automata are penalized by decreasing their current state number. If the automaton's current state is a boundary state, the related selected feature will be randomly changed. The $BestSelectedFeatures$ are returned after the last iteration of the algorithm. The single solution structure of the proposed fixed structure learning automata-based method makes it a fast and effective approach compared to the population-based optimization algorithms.

## 4 Results of experiments

The NSL-KDD [15], [16] intrusion detection dataset is employed to evaluate the proposed fixed structure learning automata-based algorithm. The Matlab 2024a software over a Macintosh OS installed MacBook Pro computer with 8 gigabytes of RAM and Intel core i5 CPU has been applied to perform the experiments.

### 4.1 The NSL-KDD intrusion detection dataset

The NSL-KDD is a well-known intrusion detection datasets, which is an updated version of the KDD99 dataset. Most of the problems in KDD99 dataset like the duplicate records and defects are updated and fixed in NSL-KDD [15], [16]. The NSL-KDD dataset includes 41 independent features for network event specifications and one dependent feature as the target class. The record types and counts in the NSL-KDD dataset are indicated in Table 1.

### 4.2 Evaluation criteria

All the methods in this paper are compared to obtain a set of selected features, which are used to build a network event classifier. Several measures like the accuracy rate, setection rate, and false positiverate can be used to evaluate the classifier, most of them are calculated using the confusion table of Table 2 [17].

Table 2: Confusion table

| | | Estimated type of current event | |
|---|---|---|---|
| | | Intrusion | Normal |
| Real type of current event | Normal | False Positive (FP) | True Negative (TN) |
| | Intrusion | True Positive (TP | False Negative (FN)) |

The first measure to evaluate the classifier is the Accuracy Rate (AR) to distinguish normal events from intrusions. The accuracy rate must be as high as possible, which can be computed by formula (1).

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

Intrusion Detection Rate (DR), which can be computed by formula (2), is the second evaluation parameter for the intrusion detection systems. The DR indicates the rate of network packets that are suspicious to be an intrusion and are correctly recognized. The False Positive Rate (FPR) is the third measure, which can be computed by formula (3). The FPR indicates the rate of network packets identified as an intrusion but is normal. A higher accuracy and detection rate and a lower false positive rate are desirable for a well-designed intrusion detection classifier.

$$DR = \frac{TP}{FN + TP} (2)$$

$$FPR = \frac{FP}{FN + TP} (3)$$

The AR of the created classifier by the selected features is employed in this paper as the fitness function for the proposed and compared algorithms. The naive Bayesian network has been applied as a simple technique to build the classifier. Several classification methods have been proposed in the literature, such as the Bayesian and neural networks [18], which can be applied to construct an intrusion detection classifier. The naive Bayesian network has been employed in this paper to build a classifier with and evaluate the selected features by the compared algorithms. The proposed fixed structure learning automata-based algorithm is compared with the improved crow search [19], particle swarm optimization [20], secretary bird optimization [21], and the genetic algorithm [5] for solving the feature selection problem. A set of integers representing the selected feature numbers constitutes the solution structure of the proposed fixed structure learning

Table 3: Obtained accuracy rate for 5, 8, 15, and 20 features

| Algorithm | Feature count 5 | 8 | 15 | 20 |
|---|---|---|---|---|
| Fixed structure learning automata | 90.31 | 91.39 | 92.57 | 90.14 |
| Crow search algorithm | 90.4 | 91.76 | 87.92 | 91.44 |
| Secretary bird optimizer | 90.79 | 89.69 | 92.46 | 91.32 |
| Particle swarm optimization | 86.91 | 88.81 | 88.31 | 91.1 |
| Genetic algorithm | 87.23 | 87.9 | 91.12 | 88.21 |

automata-base method and the genetic algorithm [5] implemented for the comparisons. The solution structure of the other compared algorithms contains 41 real values. Each value in the solution indicates the importance of the related feature, where the higher amounts are candidates for selection. The implemented algorithms in this paper evolute the solutions during the iterations by exploring the feature selection problem's search space. The features related to higher numbers in the solutions are used to build an intrusion detection classifier, which is employed to train with the training part of the intrusion detection dataset.

The next phase is classifying the test part of the intrusion detection dataset using the trained classifier. The number of selected features is also important, where on one side, the processing load of the intrusion detection system increases with more features. On the other side, the accuracy rate of the system may decrease with duplicated or unimportant features. Considering these issues, the feature selection problem needs a multi-objective algorithm, and one of the objectives must be the feature count. In this paper, for the simplicity of the performed experiments, the count of selected features is predetermined to be 5, 8, 15, and 20 features. As the results of the first experiment, the comparison of the accuracy rate for the investigated algorithms for the NSL-KDD intrusion detection dataset is presented in Table 3 and Figure 6.

Figure 6 and Table 3 demonstrate that the fixed structure learning automata-based method delivers a higher accuracy rate for 15 selected features in contrast with the existing algorithms. However, the accuracy rate of the proposed algorithm is lower than other algorithms for 5, 8, and 20 numbers of features. Nevertheless, the highest accuracy rate value is obtained from the fixed structure learning automata. Table 4 and Figure 7 demonstrate detection rates for different algorithms.

Table 4 and Figure 7 indicate that the fixed structure

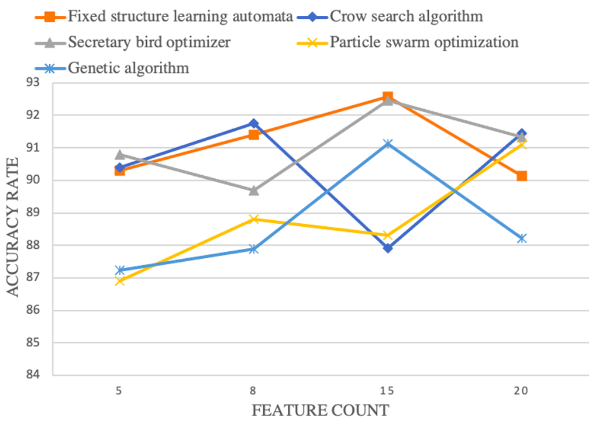

Figure 6: Compairing accuracy rates for 5, 8, 15, and 20 features

Table 4: Obtained detection rate for 5, 8, 15, and 20 features

| Algorithm | Feature count 5 | 8 | 15 | 20 |
|---|---|---|---|---|
| Fixed structure learning automata | 93.21 | 94.68 | 96.9 | 94.27 |
| Crow search algorithm | 92.65 | 94.81 | 94.24 | 93.51 |
| Secretary bird optimizer | 93.23 | 94.79 | 96.31 | 95.25 |
| Particle swarm optimization | 91.13 | 92.34 | 89.31 | 90.24 |
| Genetic algorithm | 89.9 | 91.78 | 89.13 | 89.24 |

learning automata-based method does better again for the detection rate with 15 features. For 5 features, the secretary bird optimizer algorithm results in the highest detection rate, whereas for 8 features the crow search algorithm obtains the highest value. For 20 features, as Table 4 and Figure 7 show, the secretary bird optimizer has the highest detection rate. The overall detection rate value of fixed structure learning automata-based algorithm is higher than other compared methods. Table 5 and Figure 8 depict the false positive rate value of investigated algorithms for the NSL-KDD intrusion detection dataset.

As mentioned in the previous sections, an efficient feature selection algorithm must result in a classifier with a low false positive rate in intrusion detection systems. Table 5 and Figure 8 show the desirable performance of the proposed algorithm for the false positive rate. The lowest value for 5 selected features is obtained by the secretary bird optimizer. The Crow search algorithm has the lowest false positive rate for 8 and 20 features, whereas the fixed structure learning automata-

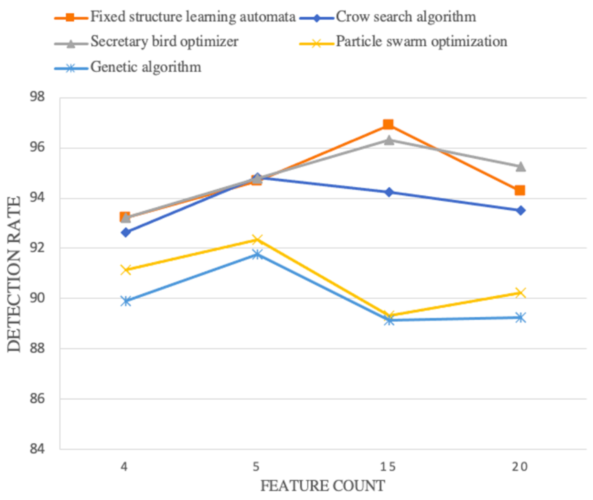

Figure 7: Compairing detection rates for 5, 8, 15, and 20 features

Table 5: Obtained false positive rate for 5, 8, 15, and 20 features

| Algorithm | Feature count 5 | 8 | 15 | 20 |
|---|---|---|---|---|
| Fixed structure learning automata | 20.04 | 16.62 | 11.37 | 15.24 |
| Crow search algorithm | 20.89 | 13.53 | 12.83 | 13.63 |
| Secretary bird optimizer | 19.84 | 15.38 | 12.51 | 14.82 |
| Particle swarm optimization | 20.51 | 15.29 | 14.09 | 14.78 |
| Genetic algorithm | 23.91 | 17.54 | 13.78 | 15.79 |

based algorithm obtains the lowest value for 15 features. The lowest value of Table 5 belongs to the proposed method, too. The obtained results of the compared algorithms in all experiments indicate that 15 is a reasonable number for the count of selected features. Also, the proposed fixed structure learning automata-based algorithm offers an acceptable performance in the experiments, where its results are competitive with the other compared methods.

## 5 Conclusion

A fixed structure-based learning automata has been introduced in this paper to tackle the feature selection problem in intrusion detection systems. The evaluations showed that the proposed method produces competitive results compared to the other methods. The reason for the new method's high performance is establishing a

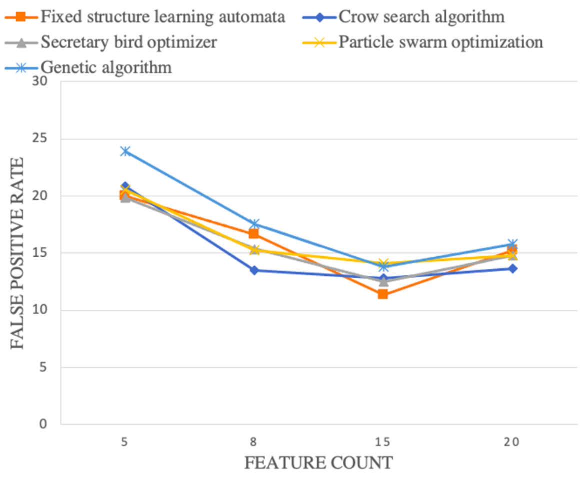

Figure 8: Compairing false positive rates for 5, 8, 15, and 20 features

proper balance between the exploration and exploration of the problem's search space. The exploitation is done with a reward for a good action or feature and a penalty for the actions that only change the related state. The exploration is performed when the penalties of an action of an automaton change the state of the action to a boundary state and the selected feature of that action changes. In this way, the set of features changes during the learning process, and the near-optimal selected set is achieved. Proposing a hybrid method of fixed structure learning automata and a discrete optimization approach can be considered as future work.

## References

[1] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," Expert Syst Appl, vol. 42, no. 5, pp. 2670–2679, Apr. 2015, doi: 10.1016/j.eswa.2014.11.009.

[2] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," Comput Secur, vol. 81, pp. 148–155, Mar. 2019, doi: 10.1016/j.cose.2018.11.005.

[3] T. Khorram and N. A. Baykan, "Feature selection in network intrusion detection using metaheuristic algorithms," International Journal Of Advance Research, Ideas and Innovations in Technolog, vol. 4, no. 4, pp. 704–710, 2018.

[4] M. H. Aghdam and P. Kabiri, "Feature Selection for Intrusion Detection System Using Ant Colony Optimization," 2016. Accessed: Jun. 06, 2022. [Online]. Available: http://ijns.jalaxy.com.tw/contents/ijns-v18-n3/ijns-2016-v18-n3-p420-432.pdf

[5] Z. Halim et al., "An effective genetic algorithm-based feature selection method for intrusion detection sys-

tems," Comput Secur, vol. 110, p. 102448, Nov. 2021, doi: 10.1016/j.cose.2021.102448.

[6] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer," Expert Syst Appl, vol. 148, p. 113249, Jun. 2020, doi: 10.1016/j.eswa.2020.113249.

[7] T. S. Naseri and F. S. Gharehchopogh, "A Feature Selection Based on the Farmland Fertility Algorithm for Improved Intrusion Detection Systems," Journal of Network and Systems Management, vol. 30, no. 3, pp. 1–27, Jul. 2022, doi: 10.1007/s10922-022-09653-9.

[8] "Learning Automata: An Introduction - Kumpati S. Narendra, Mandayam A.L. Thathachar - Google Books." Accessed: Jul. 24, 2024. [Online]. Available: https://books.google.de/books /about/LearningAutomata.html?id=ZwbCAgAAQBAJ redir-esc=y

[9] B. J. Oommen, R. O. Omslandseter, and L. Jiao, "The object migration automata: its field, scope, applications, and future research challenges," Pattern Analysis and Applications, vol. 26, no. 3, pp. 917–928, Aug. 2023, doi: 10.1007/S10044-023-01163-X.

[10] K. Asghari, M. Masdari, F. Soleimanian Gharehchopogh, and R. Saneifard, "A fixed structure learning automata-based optimization algorithm for structure learning of Bayesian networks," Expert Syst, vol. 38, no. 7, Nov. 2021, doi: 10.1111/exsy.12734.

[11] G. I. Papadimitriou, M. S. Obaidat, and A. S. Pomportsis, "On the use of learning automata in the control of broadcast networks: A methodology," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 32, no. 6, pp. 781–790, Dec. 2002, doi: 10.1109/TSMCB.2002.1049612.

[12] S. Sabamoniri, K. Asghari, and M. Javad Hosseini, "Solving Single Machine Total Weighted Tardiness Problem using Variable Structure Learning Automata," Int J Comput Appl, vol. 56, no. 1, pp. 37–42, Oct. 2012, doi: 10.5120/8858-2816.

[13] K. Asghari, A. S. Mamaghani, and M. R. Meybodi, "An evolutionary algorithm for query optimization in database," in Innovative Techniques in Instruction Technology, E-Learning, E-Assessment, and Education, Kluwer Academic Publishers, 2008, pp. 249–254. doi: 10.1007/978-1-4020-8739-4-44.

[14] S. Mukherjee and N. Sharma, "Intrusion Detection using Naive Bayes Classifier with Feature Reduction," Procedia Technology, vol. 4, pp. 119–128, Jan. 2012, doi: 10.1016/j.protcy.2012.05.017.

[15] "NSL-KDD — Datasets — Research — Canadian Institute for Cybersecurity — UNB." Accessed: Aug. 02, 2023. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[16] "NSL-KDD — IEEE DataPort." Accessed: Oct. 17, 2024. [Online]. Available: https://ieee-dataport.org/documents/nsl-kdd-0files

[17] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," Jan. 01, 2013, Academic Press. doi: 10.1016/j.jnca.2012.09.004.

[18] A. Shenfield, D. Day, and A. Ayesh, "Intelligent intrusion detection systems using artificial neural networks," ICT Express, vol. 4, no. 2, pp. 95–99, Jun. 2018, doi: 10.1016/j.icte.2018.04.003.

[19] D. Jayalatchumy, R. Ramalingam, A. Balakrishnan, M. Safran, and S. Alfarhood, "Improved Crow Search-based Feature Selection and Ensemble Learning for IoT Intrusion Detection," IEEE Access, 2024, doi: 10.1109/ACCESS.2024.3372859.

[20] A. J. Malik, W. Shahzad, and F. A. Khan, "Network intrusion detection using hybrid binary PSO and random forests algorithm," Security and Communication Networks, vol. 8, no. 16, pp. 2646–2660, Nov. 2015, doi: 10.1002/sec.508.

[21] Y. Fu, D. Liu, J. Chen, and L. He, "Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems," Artif Intell Rev, vol. 57, no. 5, pp. 1–102, May 2024, doi: 10.1007/S10462-024-10729-Y/FIGURES/4.