Journal of Mathematical Modeling
Vol. 13, No. 4, 2025, pp. 789-808. Research Article

JMM

Application of fixed structure learning automata for designing intrusion detection systems

Kayvan Asghari[†*]

[†]*Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran*
Email(s): k.asghari@iau.ac.ir

Abstract. Designing an efficient intrusion detection system involves several phases, with feature selection being one of the most important. In this paper, a fixed-structure learning automata has been applied for the feature selection phase. The introduced method includes the exploration and exploitation phases of an optimization method to find the significant features of the network events to detect intrusions. The count of the selected features in the proposed method is a pre-defined number, as the feature selection is a multi-objective problem, and one of its 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 enhances the intrusion detection accuracy rate, another significant objective for a feature selection method. Two well-known intrusion detection datasets called NSL-KDD and UNSW-NB15 have 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.
AMS Subject Classification 2010: 68T20.

1 Introduction

Various optimization algorithms have been proposed to solve optimization problems in recent years. Having several common characteristics, these algorithms are not general approaches for all optimization problems. Each of these algorithms has weaknesses that prevent it from reaching optimal solutions. One of the main issues with 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 performing a complete scan of the problem search space. However, in the last iterations, these algorithms change their strategy to search around the solutions found in the previous steps to reach the global optimal solution. The

*Corresponding author
Received: 2 February 2025 / Revised: 11 May 2025 / Accepted: 6 June 2025
DOI: 10.22124/jmm.2025.29713.2656

mentioned phases in various literature are called exploration and exploitation, respectively. 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 change during the learning process. These parameters direct the optimization algorithm to explore the problem's search space in the first iterations. But at the ending iterations and during the exploitation, they navigate the algorithm to perform a local search around the best-found solutions of 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 (IDS) is a significant task for secure networks. Various types of IDSs have been introduced so far. One of the well-known categorizations is the anomaly-based and signature-based systems, and the second category is investigated in this paper. One of the significant phases of designing an IDS is selecting the important features for checking. These features are applied to build a minimal, quick, and accurate event classifier. Various feature selection methods have been proposed in the literature so far [12]. The application of fixed-structure learning automata for solving the feature selection problem of designing an IDS has been presented in this paper. The naive Bayesian network has been employed as a classifier to evaluate the selected features of the proposed algorithm and other compared methods. During the iterations of the learning process, the automata interact with the environment and improve their actions by employing the penalty and reward mechanisms.

2 Related works

A brief explanation of the learning automata tool, its types, and the feature selection problem in developing IDSs has been presented in this section.

2.1 Feature selection in IDSs

The IDSs must spend a lot of energy and time classifying a network packet as an intrusion or a normal event. The reason is the formation of each packet from tens of features, where some of them are duplicated or useless. Researchers have introduced various feature selection methods for designing efficient and rapid IDSs [12]. Malik et al. [14] have modified the PSO algorithm to apply to feature selection in IDSs. The applied PSO algorithm has a binary solution structure and uses the random forest as the event classifier, including a group of decision trees. For every input record, all the decision trees perform the classification, and the ultimate decision of the random forest method is the decision that the majority of trees announce. The Cuttlefish optimization algorithm has been employed by Eesa et al. [7] for feature selection, where some decision trees have been used as event classifiers and KDD-Cup99 as the dataset. Selvakumar and Muneeswaran [27] have applied the Firefly algorithm for the feature selection process in IDSs. They have employed the KDD-Cup99 dataset. Acharya and Singh [1] have introduced a feature selection method for IDS using the Intelligent Water Drops (IWD) optimization algorithm. They employed the Support Vector Machine (SVM) as a classifier to evaluate the selected feature subset by the IWD. The applied IDS dataset has been the KDD-CUP99. A wrapper-based feature selection method to build IDSs has been introduced by Alazzam et al. [2], which applies the pigeon-inspired optimization

algorithm. They investigated two binary revisions of their algorithm, where UNSW-NB15, NSL-KDD, and KDD-Cup99 are employed as the intrusion detection datasets. Naseri and Gharehchopogh [20] have modified the Farmland Fertility algorithm to a binary algorithm for selecting features in IDSs. The NSL-KDD and UNSW-NB15 have been applied as datasets, and SVM, decision tree, AdaBoost, random forest, K-Nearest Neighbor (KNN), and naive Bayesian have been investigated as intrusion classifiers.

Pan et al. [24] have introduced an improved Gray Wolf Optimization (GWO) method to select optimized feature subsets in high dimensional datasets and fix the stagnation problem of GWO. The introduced algorithm improves the initialization phase of the search process for feature selection in IDSs. Ghanbarzadeh et al. [9] have presented the application of the Horse herd Optimization Algorithm (HOA) for selecting an effective feature subset to build a network IDS. They have discretized and binarised the HOA and developed a quantum-inspired HOA to improve the horse herd's social behaviors. The resulting algorithm has been transformed into a multi-objective method and employed for feature selection. The KNN classifier has also been applied to evaluate the selected features using the NSL-KDD and CSE-CIC-IDS2018 intrusion detection datasets. Jayalatchumy et al. [11] have used an improved Crow search algorithm to select features in IDSs. They have applied some data-denoising approaches to eliminate imbalances and deficiencies in intrusion detection datasets. An ensemble classifier for evaluating the features has also been employed with the UNSW-NB15 and NSL-KDD datasets.

2.2 The learning automaton

A learning automaton is an abstract learning tool, implementable in different ways and usable 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 various classifications for learning automata, one of which is the fixed and variable structures. The probability of changing actions and transitions between the automaton's states is a fixed value in the fixed structure. In contrast, in the variable structure learning automata, the mentioned probabilities are updated considering the environment's feedback [19]. The fixed-structure learning automata are convenient for solving problems with small and well-defined search spaces. It has low computational overhead and requires less memory and simpler hardware to implement, which makes it appropriate for IoT devices or microcontrollers with limited processing power and resource constraints. Fixed-structure learning automata have been used to solve many optimization problems like the join ordering problem of database queries [15], the total weighted tardiness scheduling problem [5], the Hamiltonian cycle problem [6], and the software clustering problem. The common characteristic of these problems is a solution structure containing a permutation of elements. New solutions are obtained by selecting each element from a limited number of elements. The selection of different permutations is performed by using the actions of the fixed-structure learning automata. For example, each member of the solution array is a join operator in database queries for the join ordering problem. The elements are several jobs that are scheduled to execute in the total weighted tardiness scheduling problem. The fixed-structure learning automata have also been used as chromosomes of the genetic algorithm for several optimization problems, resulting in a hybrid optimization algorithm [16]. They make decisions quickly and efficiently and manage the balance of controlled random exploration and deterministic exploitation through their reward and penalty mechanisms. The fixed-structure learning automata can obtain a permutation of the best features

to solve the feature selection problem in intrusion detection systems and build a near-optimal classifier of network events. The object migration learning automata, as a fixed-structure type, have been used in this paper to solve the feature selection problem in IDSs [23].

The learning automaton is defined formally by the quintuple of $\{\phi, \alpha, \beta, F(\cdot,\cdot), H(\cdot,\cdot)\}$, where ϕ is a set of internal states ($\phi = \{\phi_1, \phi_2, ..., \phi_s\}$), α is a set of automaton actions ($\alpha = \{\alpha_1, \alpha_2, , \alpha_r\}$), and β is a set of environment responses ($\beta = \{\beta_1, \beta_2, , \beta_m\}$). The finite-type state-output automata have been employed in this paper to select features in IDS because the ϕ, α, and β sets are finite. The state, action, and response of the environment for the learning automaton at 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. The 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 ($F(\cdot,\cdot) : \phi \times \beta \rightarrow \phi$). Finally, $H(\cdot,\cdot)$ is a function for determining the current action with the current environment response and current state ($H(\cdot,\cdot) : \phi \times \beta \rightarrow \alpha$)) [19]. 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 automata with the working environment is illustrated in Figure 1.

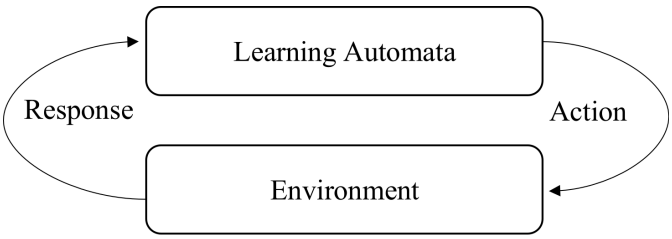

Figure 1: The interaction between the learning automata and the environment.

The automaton selects a feature as its action (α_i), which plays the role of environment input. For every action, the environment returns feedback (β_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 (ϕ_i). The probability of selected action (α_i) increases by the learning automaton for a reward from the environment. Instead, for a penalty response, the selected action's probability is decreased [19].

Learning automata have various applications in different fields like pattern recognition, neural and Bayesian network optimization [3], network routing [25], job scheduling [26], query optimization [4], data compression, and solving NP problems.

3 The fixed-structure learning automata for feature selection

A fixed-structure type of learning automata tool is proposed in this section to find a near-optimal solution for the feature selection problem in designing IDSs. The proposed method uses the penalty and reward mechanisms to obtain a selected feature set, which is employed to build an IDS classifier. The selected feature set represents all features in the network packet, which is minimal and has no duplicate information. Thus, the classifier constructed using the selected features can inspect the input network packet to identify the intrusions from normal network events. The naive Bayesian network [18] 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. Artificial neural networks and other classifiers can also be used to evaluate the solution, but this paper focuses on the feature selection phase. The flowchart in Figure 2 demonstrates the working procedure of the proposed method. To use an intrusion detection dataset, a pre-processing of features to convert named fields to numbers and discretize the continuous field values is necessary, which has been performed by the Weka software.

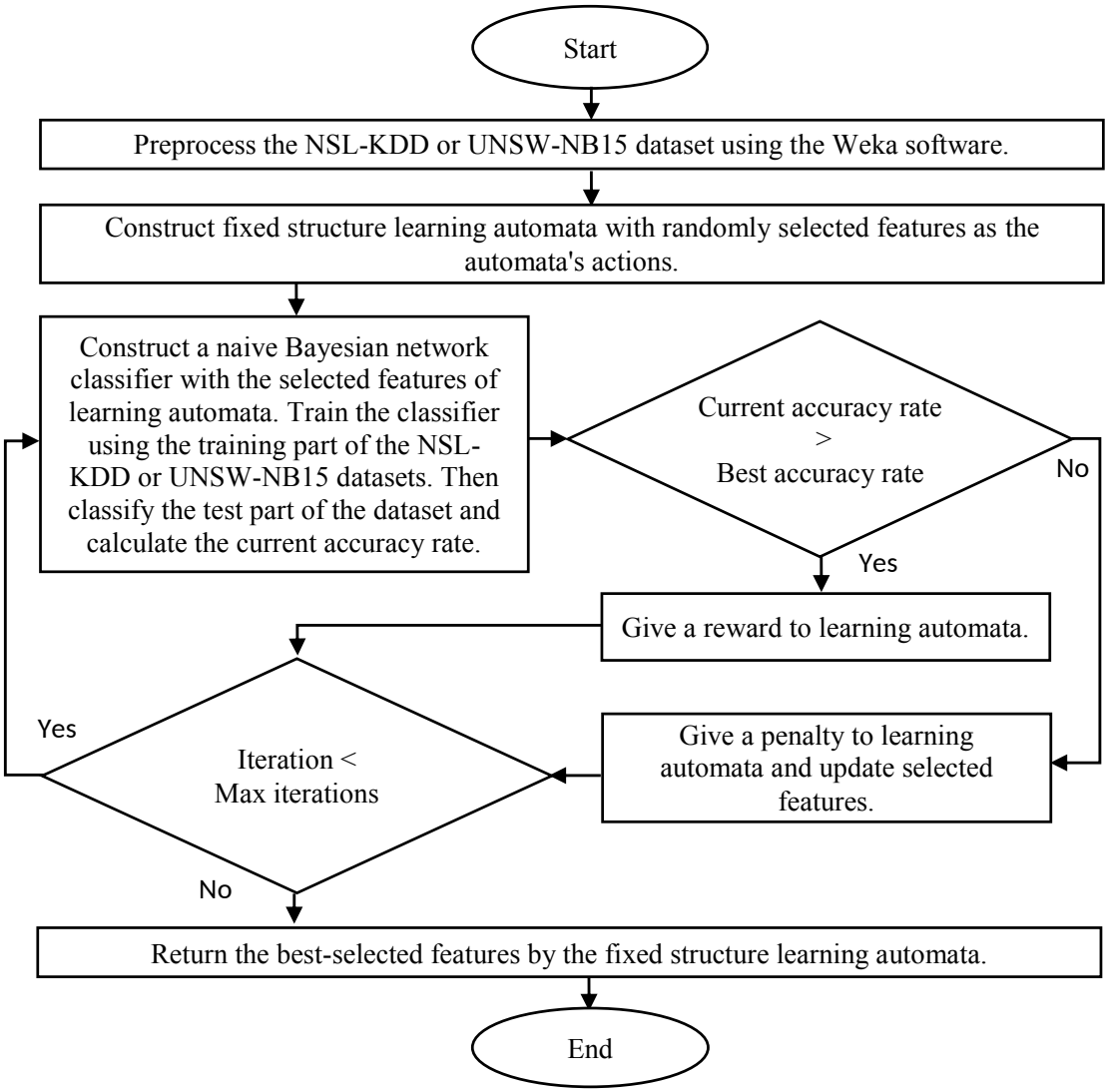

Figure 2: The fixed structure learning automata-based model for the feature selection problem.

3.1 Solution representation for the proposed algorithm

A demonstration of the fixed-structure learning automata for selecting eight features has been presented in Figure 3. The n learning automata are needed for the selection of n features, 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 changed, which causes the selection of another feature from the feature set. The action change 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 [23]. A random feature is selected and dedicated as the action of each automaton at the beginning of the proposed algorithm, where the action has a boundary state at the beginning. The states 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 increases compared to the previous iteration, the learning automata is rewarded by reducing the state number of the current actions. Figure 4 illustrates the rewarding mechanism for the actions of the learning automata of Figure 3.

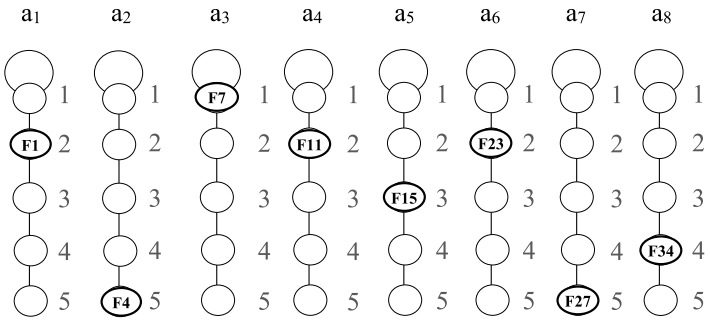

Figure 3: The fixed-structure learning automata for feature selection.

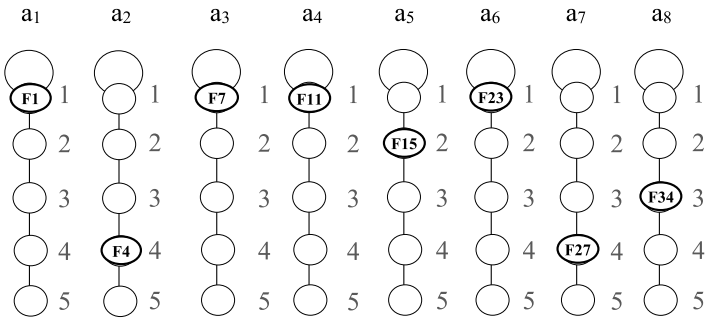

Figure 4: The reward for fixed-structure learning automata for feature selection.

If the accuracy rate of the classifier decreases compared to the previous iteration, the learning automata is penalized by increasing the state number of the current actions. 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, have been replaced with $F6$ and $F25$ features in Figure 5 by the penalty mechanism.

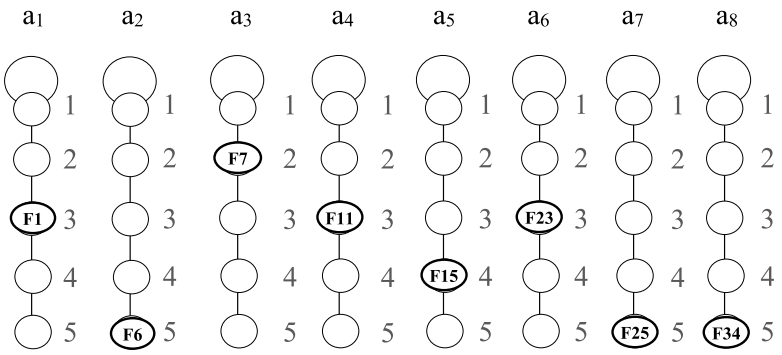

Figure 5: The penalty for fixed-structure learning automata for feature selection.

3.2 Implementation of the proposed algorithm

The fixed structure learning automata-based algorithm is a global search, which includes the exploration and exploitation phases. In the initial iterations of the proposed algorithm, the exploration phase is performed, where all the actions of automata or selected features are in the boundary state and may change frequently. After some iterations and finding some promising solutions, the state numbers of selected features are decreased due to the rewards from the environment. 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 pseudo-code of the introduced approach for the feature selection problem of IDS is presented in Algorithm 1.

In Algorithm 1, an integer array for the currently selected features, called SF, is created 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 number of the algorithm. In the first for loop of the algorithm, the learning automata select the features related 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 of the intrusion detection dataset 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, learning automata are penalized by decreasing their current state numbers. If the automaton's current state is a boundary one, the related selected feature is randomly changed. The $BestSelectedFeatures$ are returned after the last iteration of the algorithm. The single solution structure of the proposed method makes it a fast and effective approach compared to the population-based optimization algorithms.

Implementing and using the fixed structure learning automata-based feature selection algorithm in real-world IDSs has several advantages. This algorithm can be used in IDSs to process network packets in real-time. The learning automata continuously adjust the decision-making process based on live feedback (e.g., from attack labels, system administrator feedback, or system responses). Few data structures and related functions are required to implement the fixed-structure learning automata tool. Given the simplicity, lightweight nature, and low resource usage of the learning automata-based tool, the proposed algorithm can also work well on low-power software and hardware platforms without significantly reduc-

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 $FSLA_i$ (ith Fixed-Structure Learning Automaton, where $1 \leq i \leq$ Count of Selected Features);
$AR_{Best} = 0$; $Iter = 1$;
Define BestSelectedFeatures[i]; ($1 \leq i \leq$Count Of Selected Features);
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**
        {Rewarding automata}
        $AR_{Best} = AR_{Current}$;
        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
        {Penalizing the automata}
        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;

ing the system performance. This characteristic also makes such a system independent of cloud systems. The proposed algorithm can be used in host-based and network-based IDSs. A network event classifier is built from the selected features by the proposed algorithm. When the relevant event is recognized as a true positive attack, the learning automata actions are rewarded, which indicates that the appropriate

features are selected. If a false positive attack has occurred or an attack has occurred but has not been detected, the actions of the automata are penalized. To do that, by changing the states of each action, the selected features are changed in subsequent iterations, and the classifier structure is reconstructed. Another advantage of using the fixed-structure learning automata tool in IDSs is that it does not necessarily need a large-sized offline training dataset before starting to work. Of course, using such a dataset, as has been done in the experiments of this paper, improves the system performance even in the initial iterations and increases the system accuracy. On the other hand, at the beginning of the proposed algorithm, the convergence of the algorithm to the optimal selected features may be slow, and this is due to the fixed structure of the learning automata actions for selecting features.

3.3 The computational complexity of the proposed algorithm

The computational complexity of the proposed algorithm has been evaluated in this section. The proposed algorithm has been compared with several metaheuristic algorithms, most of which have limited memory consumption. Considering the use of a population of solutions in the compared algorithms and only one solution for the proposed one, it is clear that the memory consumption of the proposed algorithm is lower than that of the others. However, regarding time complexity, different algorithms have different behaviors, and algorithms with lower computational complexity perform better. For the proposed algorithm, the number of automata actions or the number of selected features (n), the time to create the classifier, the time to train the classifier, the time to classify events, and the maximum number of iterations (t) are involved in the computational complexity. In the first iteration of the proposed algorithm, a set of fixed-structure learning automata is created, which has the computational complexity of $O(n)$. For each automaton's action, a feature is selected randomly, and the selected features by the algorithm must be used and evaluated to build an event classifier. Creating a nave Bayesian classifier has the computational complexity of $O(m \times n)$ for m training records. For classifying events with k test records and two attack and non-attack classes for events, there is a time complexity of $O(k \times 2 \times n)$, equal to $O(k \times n)$. In each iteration of the proposed algorithm, after building the classifier using the selected features, if the accuracy rate of the classifier decreases compared to previous iterations, the automata actions are penalized with complexity $O(n)$. If an action reaches its boundary state, it randomly changes its selected feature with complexity $O(n)$. The newly selected feature must be rechecked to ensure that other automata actions havenot chosen it before. Therefore, the maximum time complexity of the penalty operation is $2 \times O(n)$, equal to $O(n)$. If the classifier accuracy rate increases, the automata actions with complexity $O(n)$ are rewarded, stabilizing the selected features for the subsequent iterations. Eq. (1) shows the time complexity of the feature selection algorithm based on the fixed-structure learning automata.

$$\begin{aligned} \textit{Time Complexity} =& O(t \times [O(\textit{Constructing automata}) + O(\textit{Constructing a classifier and training it}) \\ & + O(\textit{Classifying events}) + O(\textit{Penalizing or rewarding actions of automata})]) \\ \textit{Time Complexity} =& O(t \times [O(n) + O(m \times n) + O(k \times n) + O(n)]) \\ =& O((t \times n) + (t \times n \times m) + (t \times k \times n) + (t \times n)) \\ =& O((2tn) + tn(m+k)) = O(tn(m+k)) \end{aligned} \tag{1}$$

4 Results of experiments

An attack is a situation where attackers exploit weaknesses in a computer network to gain unauthorized access, disrupt services, steal data, or cause other damage. The proposed algorithm has been evaluated using the NSL-KDD [21], [22], and UNSW-NB15 [17] datasets, where the NSL-KDD is an improved and newer version of KDD CUP99. These two datasets include attacks like DOS, probing, user to root, remote to local, worms, Shellcode, Reconnaissance, Analysis, Fuzzers, Exploits, etc. In the proposed approach of this paper, the attack detection is performed by a classifier built with the features found by the proposed algorithm, and the final intrusion detection system is signature-based. Thus, the proposed method of this paper could be employed for any identifiable attack type by a pattern of feature values. The Matlab 2024a software over a Macintosh OS installed MacBook Pro computer with 8 gigabytes of RAM and Intel core i5 CPU have been applied to perform the experiments.

4.1 The NSL-KDD intrusion detection dataset

The NSL-KDD is a well-known intrusion detection dataset, an updated version of the KDD99 dataset. Most of the problems in the KDD CUP99 dataset like the duplicate records and defects are updated and fixed in NSL-KDD. This 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.

Table 1: Content of the 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

4.2 The UNSW-NB15 intrusion detection dataset

The second well-known intrusion detection dataset, which has been used in the evaluations of this paper and contains newer attack types, is the UNSW-NB15. This dataset has nine attack types, and each record includes 42 features. Table 2 demonstrates the content details of the UNSW-NB15 dataset.

4.3 Evaluation criteria

All methods in this paper have been compared to obtain a set of selected features to build a network event classifier. Several measures, such as the accuracy rate, detection rate, and false positive rate, can be used to evaluate the classifier; most of them are calculated using the confusion table of Table 3 [13]. The first measure for evaluating the classifier is the Accuracy Rate (AR), which distinguishes normal events from intrusions. The AR must be as high as possible, which can be computed by Eq. (2). Intrusion Detection Rate (DR), which can be calculated by Eq. (3), is the second evaluation parameter for the

Table 2: Content of the UNSW-NB15 dataset.

Attack types	Train records	Test records
DOS	12264	4089
Generic	18871	40000
Worms	130	44
Shellcode	1133	378
Reconnaissance	10491	3496
Analysis	2000	677
Backdoors	1746	583
Fuzzers	18184	6062
Exploits	33393	11132
Normal	56000	37000
Total count	175341	82332

Table 3: 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)

IDSs. The DR indicates the rate of network packets that are suspicious of being an intrusion and are correctly recognized. The False Positive Rate (FPR) is the third measure, which can be computed by Eq. (4). The FPR indicates the rate of network packets identified as an intrusion, but related to normal events. A well-designed intrusion detection classifier should achieve high accuracy and detection rates, along with a low false positive rate.

$$AR = \frac{TN+TP}{TP+FN+FP+TN} \quad (2) \qquad DR = \frac{TP}{FN+TP} \quad (3) \qquad FPR = \frac{FP}{FN+TP} \quad (4)$$

The AR of the created classifier by the selected features has been employed in this paper as the fitness function for the proposed and compared algorithms. Several classification methods have been proposed in the literature, such as Bayesian and neural networks [28], which can be applied to construct an intrusion detection classifier. As a simple technique, the naive Bayesian network has been employed in this paper to build a classifier and evaluate the selected features of the compared algorithms. The proposed fixed structure learning automata-based algorithm has been compared with the improved crow search [11], particle swarm optimization [14], secretary bird optimization [8], and the genetic algorithm [10] for solving the feature selection problem. A set of integers representing the selected feature numbers constitutes the solution structure of the proposed and the genetic algorithm [10] implemented for the comparisons. The solution structure of the other compared algorithms contains a set of real numbers, where each number in the solution indicates the importance of the related feature, and the higher amounts are candidates for selection. The implemented algorithms in this paper evolve the solutions dur-

ing the iterations by exploring the feature selection problems 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 significant, where on one side, the processing load of the IDS increases with more features. On the other side, the AR 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. The count of selected features has been predetermined to be 5, 8, 15, and 20, for the simplicity of the performed experiments in this paper. The count of search agents for the compared algorithms has been 20, where 500 iterations for the executions of the algorithms have been considered. In the first experiment, the convergence of the proposed learning automata-based method to the best-obtained results in the first 100 iterations is illustrated in Figure 6 for the records of the NSL-KDD dataset.

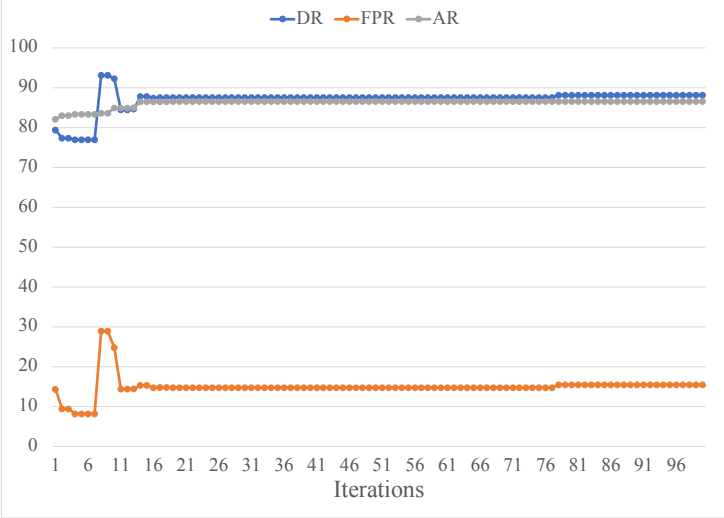

Figure 6: Convergence diagram of the proposed method for the NSL-KDD dataset.

The charts of Figure 6 indicate that the proposed algorithm explores the problem search space in the early iterations with sudden moves. In contrast, in the last iterations, the algorithm tries to converge to the best-found solutions of the previous iterations. Thus, the proposed algorithm effectively balances the exploration and exploitation phases. As a result of the second experiment, the comparison of the AR for the investigated algorithms on the NSL-KDD intrusion detection dataset has been presented in Table 4 and Figure 7. The results demonstrate that the fixed structure learning automata-based method delivers a higher AR for 15 selected features in contrast with the existing algorithms. However, the accuracy rate of the proposed algorithm is lower than the others for 5, 8, and 20 features. Nevertheless, the proposed algorithm has obtained the highest accuracy rate.

Table 4: Obtained AR values for 5, 8, 15, and 20 features of the NSL-KDD dataset.

	Feature count			
Algorithm	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

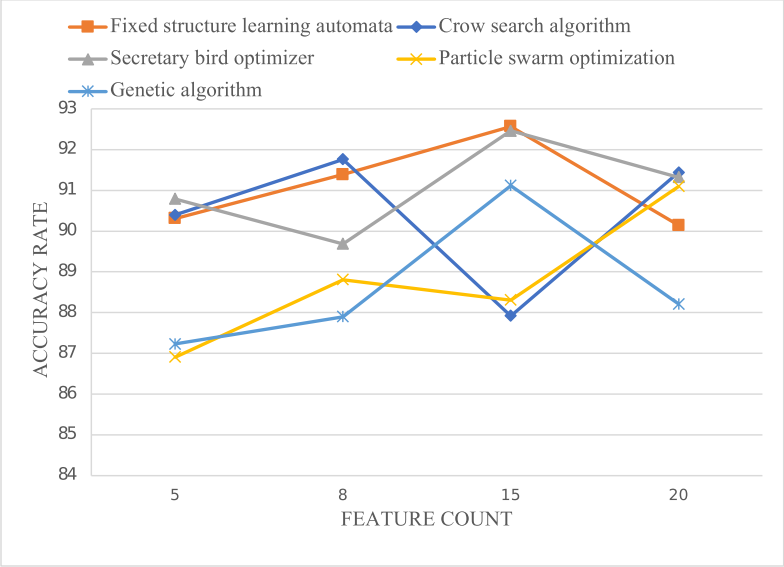

Figure 7: Compairing AR values for 5, 8, 15, and 20 features of the NSL-KDD dataset.

The next experiment has been performed on the UNSW-NB15 dataset to compare the AR of the investigated algorithms. Table 5 and Figure 8 present the results. For five features of the UNSW-NB15 dataset records, the proposed method obtains an AR lower than the crow search and the secretary bird optimizer algorithm. For eight features, the algorithm's results are lower than the secretary bird and higher than the other algorithms. However, the obtained AR of the learning automata-based algorithm is the highest value for 15 and 20 selected features.

Table 5: Obtained AR values for 5, 8, 15, and 20 features of the UNSW-NB15 dataset.

	Feature count			
Algorithm	5	8	15	20
Fixed-structure learning automata	89.9	91.5	93.74	92.92
Crow search algorithm	91.08	90.49	93.63	92.62
Secretary bird optimizer	90.45	92.3	89.98	91.15
Particle swarm optimization	88.56	88.61	92.58	90.38
Genetic algorithm	87.11	89.72	89.52	91.82

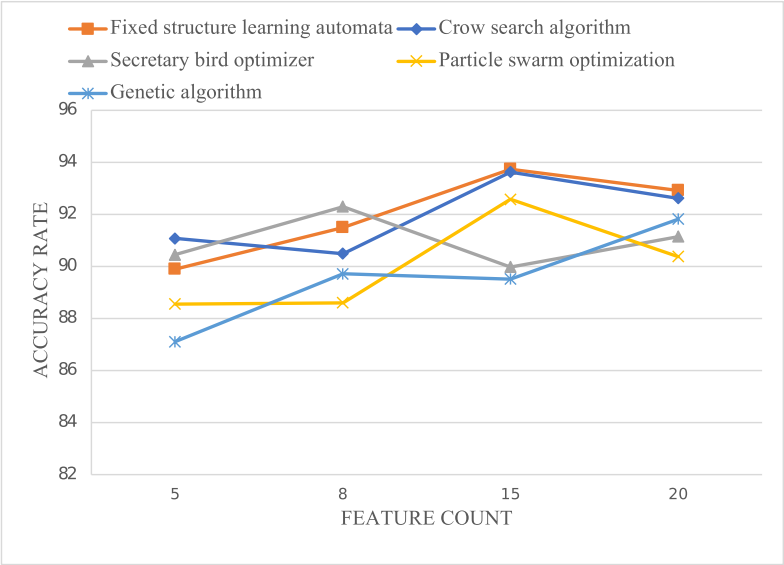

Figure 8: Compairing AR values for 5, 8, 15, and 20 features of the UNSW-NB15 dataset.

Table 6 and Figure 9 demonstrate the DR of the compared algorithms for the NSL-KDD dataset. They indicate that the fixed structure learning automata-based method does better again for the DR. The secretary bird optimizer algorithm obtains the highest DR for five features, whereas the crow search algorithm does the same for eight features. For 20 features, as Table 6 and Figure 9 show, the secretary bird optimizer has the highest DR value. The DR value of the fixed structure learning automata-based algorithm, as the maximum value of Table 6, is higher than the others when the feature count is 15.

The obtained DR values of the compared algorithms for the UNSW-NB15 dataset are presented in Table 7 and Figure 10. Again, for eight and fifteen selected features of the UNSW-NB15 dataset, the proposed algorithm exhibits the highest performance for DR, as Figure 10 illustrates. For the five and twenty selected features, the DR values of the crow search and the secretary bird optimizer are the highest, respectively. The obtained results for the PSO and genetic algorithm indicate their lower performance for DR.

Table 6: Obtained DR values for 5, 8, 15, and 20 features of the NSL-KDD dataset.

	Feature count			
Algorithm	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

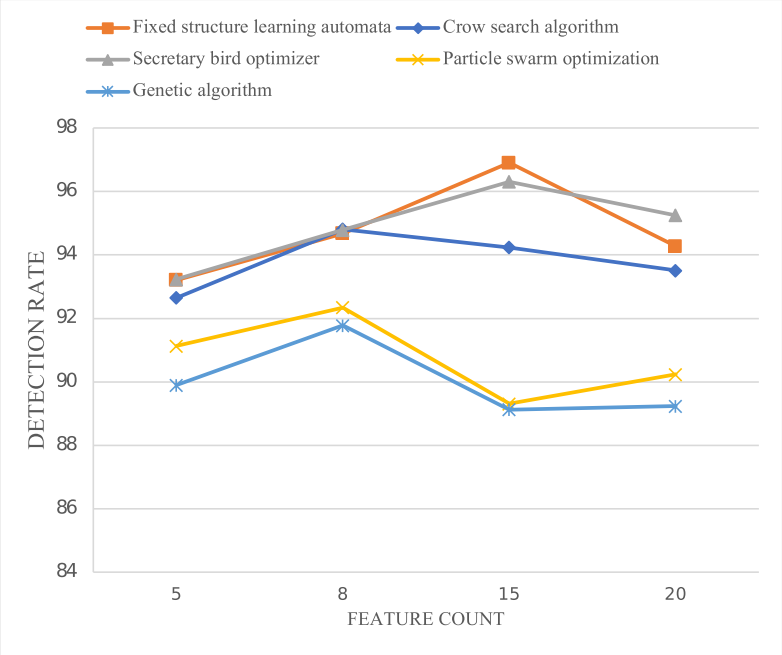

Figure 9: Compairing DR values for 5, 8, 15, and 20 features of the NSL-KDD dataset.

Table 7: Obtained DR values for 5, 8, 15, and 20 features of the UNSW-NB15 dataset.

	Feature count			
Algorithm	5	8	15	20
Fixed-structure learning automata	92.71	94.92	97.26	94.83
Crow search algorithm	93.48	94.41	94.83	93.32
Secretary bird optimizer	92.34	93.78	97.08	95.1
Particle swarm optimization	89.11	91.82	90.31	91.49
Genetic algorithm	91.25	92.35	91.47	91.38

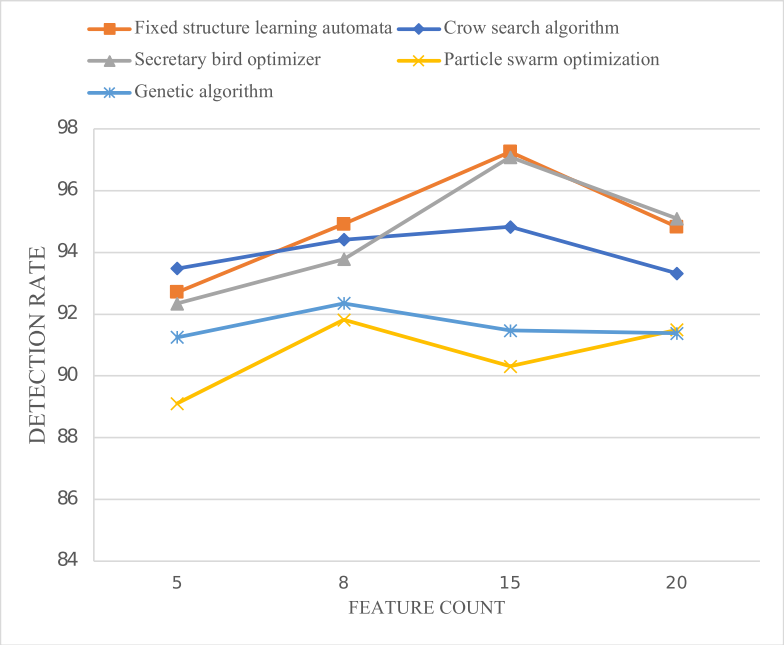

Figure 10: Compairing DR values for 5, 8, 15, and 20 features of the UNSW-NB15 dataset.

Table 8 and Figure 11 depict the FPR value of the 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 FPR value in IDSs. Table 8 and Figure 11 show the desirable performance of the proposed algorithm for the FPR value. The secretary bird optimizer has obtained the lowest value for the five selected features. The crow search algorithm has the lowest FPR for 8 and 20 features, whereas the fixed structure learning automata-based algorithm has obtained the lowest value for 15 features. The lowest value of Table 8 belongs to the proposed method, too.

The FPR values of the compared methods for the UNSW-NB15 dataset have been presented in Table 9 and Figure 12. For the UNSW-NB15 dataset, the lowest FPR value has been obtained by the crow search algorithm for five and eight selected features. For fifteen selected features, the lowest FPR value has been achieved by the learning automata-based algorithm, where the secretary bird optimizer has the lowest value for twenty features.

The results of the compared algorithms in all experiments indicate that 15 is a reasonable number for the count of selected features. The proposed fixed structure learning automata-based algorithm has offered an acceptable performance in the experiments for 15 selected features, where its results are competitive with the other compared methods.

Table 8: Obtained FPR values for 5, 8, 15, and 20 features of the NSL-KDD dataset.

	Feature count			
Algorithm	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

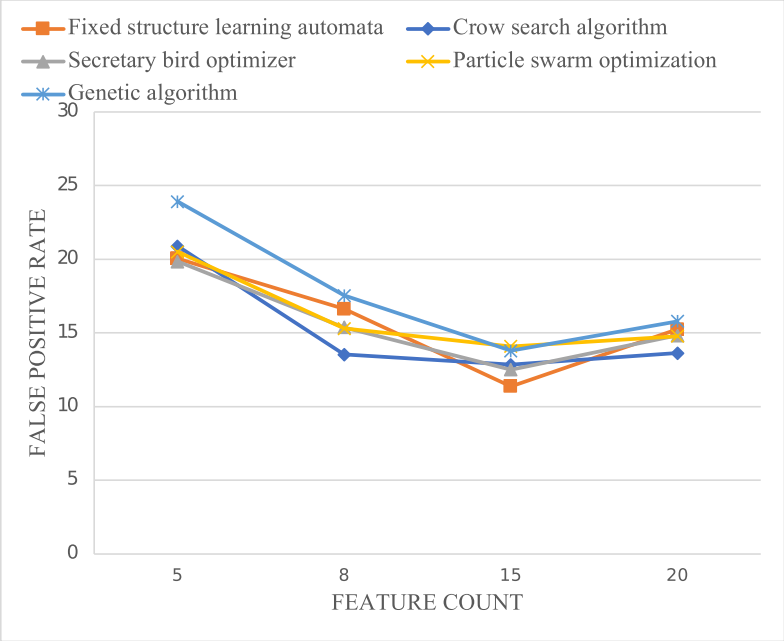

Figure 11: Compairing FPR values for 5, 8, 15, and 20 features of the NSL-KDD dataset.

Table 9: Obtained FPR values for 5, 8, 15, and 20 features of the UNSW-NB15 dataset.

	Feature count			
Algorithm	5	8	15	20
Fixed-structure learning automata	19.92	17.24	10.96	16.38
Crow search algorithm	18.75	14.81	11.91	14.83
Secretary bird optimizer	20.61	16.36	14.67	13.47
Particle swarm optimization	19.73	15.29	12.74	15.34
Genetic algorithm	24.28	17.93	14.02	14.69

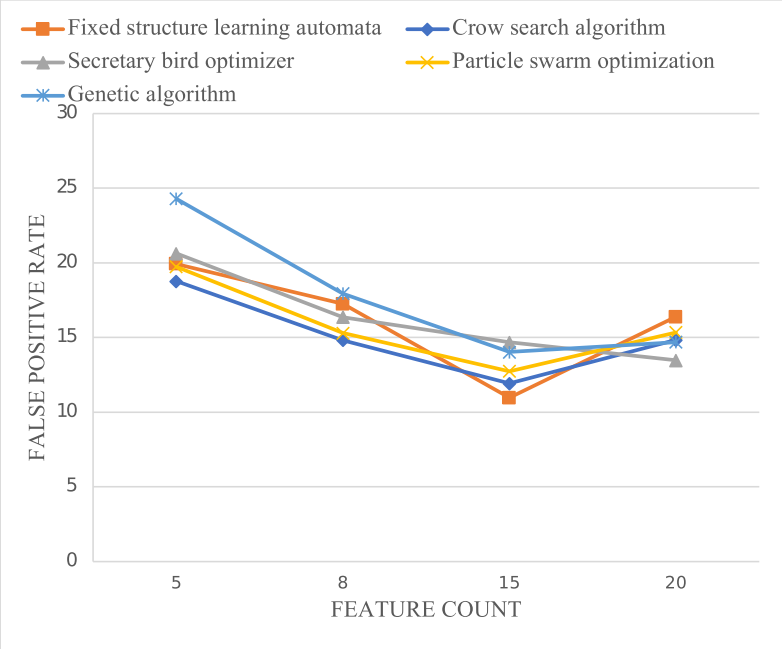

Figure 12: Compairing FPR values for 5, 8, 15, and 20 features of the UNSW-NB15 dataset.

5 Conclusion

A fixed structure learning automata-based algorithm has been introduced in this paper to tackle the feature selection problem in IDSs. The evaluations have shown that the proposed method produces competitive results compared to the other methods. The proposed method's high performance is due to establishing a proper balance between exploration and exploitation in the problem's search space. The exploitation is done with only a state change as a reward or penalty for an action. 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 of features is achieved. Proposing a hybrid method of fixed-structure learning automata and a discrete optimization approach can be considered as future work of this paper.

References

[1] N. Acharya, S. Singh, *An IWD-based feature selection method for intrusion detection system*, Soft Comput. **22** (2018) 4407–4416.

[2] H. Alazzam, A. Sharieh, K.E. Sabri, *A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer*, Expert Syst. Appl. **148** (2020) 113249.

[3] K. Asghari, M. Masdari, F.S. Gharehchopogh, R. Saneifard, *A fixed structure learning automata-based optimization algorithm for structure learning of Bayesian networks*, Expert Syst. **38** (2021) e12734.

[4] K. Asghari, A.S. Mamaghani, 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, 249–254.

[5] K. Asghari, M.R. Meybodi, *Solving single machine total weighted tardiness scheduling problem using learning automata and combination of it with genetic algorithm*, 3rd Iran Data Mining Conference, Tehran, Iran, 2009.

[6] K. Asghari, M.R. Meybodi, *Searching for Hamiltonian cycles in graphs using evolutionary methods.*, 2nd Joint Congress on Fuzzy and Intelligent Systems, Isfahan, Iran, 2008.

[7] A.S. Eesa, Z. Orman, A.M.A. Brifcani, *A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems*, Expert Syst. Appl. **42** (2015) 2670–2679.

[8] Y. Fu, D. Liu, J. Chen, L. He, *Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems*, Artif. Intell. Rev. **57** (2024) 123.

[9] R. Ghanbarzadeh, A. Hosseinalipour, A. Ghaffari, *A novel network intrusion detection method based on metaheuristic optimisation algorithms*, J. Ambient Intell. Humaniz. Comput. **14** (2023) 7575–7592.

[10] Z. Halim, M.N. Yousaf, M. Waqas, M. Sulaiman, G. Abbas, M. Hussain, I. Ahmad, M. Hanif, *An effective genetic algorithm-based feature selection method for intrusion detection systems*, Comput. Secur. **110** (2021) 102448.

[11] D. Jayalatchumy, R. Ramalingam, A. Balakrishnan, M. Safran, S. Alfarhood, *Improved Crow Search-based Feature Selection and Ensemble Learning for IoT Intrusion Detection*, IEEE Access **12** (2024) 33218–33235.

[12] T. Khorram, N.A. Baykan, *Feature selection in network intrusion detection using metaheuristic algorithms*, Int. J. Advance Research, Ideas and Innovations in Technology, **4** (2018) 704–710.

[13] H.J. Liao, C.H.R. Lin, Y.C. Lin, K.Y. Tung, *Intrusion detection system: A comprehensive review*, J. Netw. Comput. Appl. **36** (2013) 16–24.

[14] A.J. Malik, W. Shahzad, F.A. Khan, *Network intrusion detection using hybrid binary PSO and random forests algorithm*, Security Comm. Networks. **8** (2015) 2646–2660.

[15] A.S. Mamaghani, K. Asghari, F. Mahmoudi, M.R. Meybodi, *A novel hybrid algorithm for join ordering problem in database queries*, Proceedings of 6th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics, Tenerife, Spain, 2007.

[16] A.S. Mamaghani, K. Asghari, M.R. Meybodi,*Designing a new structure based on learning automaton to improve evolutionary algorithms (with considering some case study problems).*, J. Adv. Comput. Res. **4** (2013) 1–24.

[17] N. Moustafa, J. Slay, *UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)*, Military Communications and Information Systems Conference (MilCIS), IEEE, 2015.

[18] S. Mukherjee, N. Sharma, *Intrusion detection using naive Bayes classifier with feature reduction*, Proc. Technol. **4** (2012) 119–128.

[19] K.S. Narendra, M.A.Thathachar, *Learning Automata: An Introduction*, Dover Publications, 2013.

[20] T.S. Naseri, F.S. Gharehchopogh, *A feature selection based on the farmland fertility algorithm for improved intrusion detection systems*, J. Netw. Syst. Manag. **30** (2022) 40.

[21] *NSL-KDD dataset*, Canadian Institute for Cybersecurity, Available: https://www.unb.ca/cic/datasets/nsl.html, 2009.

[22] *NSL-KDD*, IEEE DataPort, Available: https://ieee-dataport.org/documents/nsl-kdd-0, 2022.

[23] B.J. Oommen, R.O. Omslandseter, L. Jiao, *The object migration automata: its field, scope, applications, and future research challenges*, Pattern Anal. Appl. **26** (2023) 917–928.

[24] H. Pan, S. Chen, H. Xiong, *A high-dimensional feature selection method based on modified Gray Wolf Optimization*, Appl. Soft Comput. **135** (2023) 110031.

[25] G.I. Papadimitriou, M.S. Obaidat, A.S. Pomportsis, *On the use of learning automata in the control of broadcast networks: A methodology*, IEEE Trans. Syst. Man Cybern. B Cybern. **32** (2002) 781–790.

[26] S. Sabamoniri, K. Asghari, M.J. Hosseini, *Solving single machine total weighted tardiness problem using variable structure learning automata*, Int. J. Comput. Appl. **56** (2012) 37–42.

[27] B. Selvakumar, K. Muneeswaran, *Firefly algorithm based feature selection for network intrusion detection*, Comput. Secur. **81** (2019) 148–155.

[28] A. Shenfield, D. Day, A. Ayesh, *Intelligent intrusion detection systems using artificial neural networks*, ICT Express **4** (2018) 95–99.