Applied Soft Computing Journal 170 (2025) 112416

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Asymmetric variable depth learning automaton and its application in defending against selfish mining attacks on bitcoin

Ali Nikhalat-Jahromi *, Ali Mohammad Saghiri, Mohammad Reza Meybodi

Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Keywords:
Learning Automata
Reinforcement Learning
Hybrid Learning Models
Fixed Structure Learning Automata
Variable Action Set Learning Automata
Selfish Mining
Bitcoin
Recommendation Systems

ABSTRACT

Learning Automaton (LA), a branch of reinforcement learning, initially began with the Fixed Structure Learning Automaton (FSLA) family and was later expanded to include the Variable Structure Learning Automaton (VSLA) family. Tuning the depth of the FSLA in complex environments has long been a challenging task, significantly limiting the ability to effectively navigate the exploration–exploitation dilemma. No solution has been found for this open problem yet. This study addresses this issue by introducing a novel hybrid learning automaton model called Asymmetric Variable Depth Hybrid Learning Automaton (AVDHLA). The AVDHLA model intelligently learns the depth of fixed structure LA in an autonomous manner by combining the $L_{KN,K}$ from the FSLA class and Variable Action Set LA (VASLA) from the VSLA class. Computer simulations are conducted to validate the proposed model in diverse environments, including both stationary and non-stationary (Markovian switching and State-dependent) scenarios. Performance evaluation is based on predefined metrics, such as total number of rewards (TNR) and action switching (TNAS). Statistical tests indicate that across both stationary and non-stationary environments, the AVDHLA consistently outperforms the $L_{KN,K}$ in terms of TNR and TNAS across the majority of experiments. Moreover, the AVDHLA model is applied in two key applications. Firstly, it is used to defend against the selfish mining attack in Bitcoin and is compared with the well-known tie-breaking mechanism. Simulation results consistently demonstrate that our proposed method increases the threshold for successful selfish mining attacks from 25% to 40%. Secondly, the AVDHLA model has been applied to develop a novel learning automaton-based recommendation system. The results demonstrate the superiority of the proposed method in terms of the Click-Through Rate (CTR) and Precision compared to previous approaches.

1. Introduction

A learning automaton (LA) is a self-adaptive reinforcement learning model [1–12] that enhances its performance through continuous interaction with a random environment. It learns to select the optimal action based on feedback (response) received from the environment, which can be either favorable or unfavorable [13].

The inception of the learning automaton can be traced back to Tsetlin's pioneering work [14], which introduced both deterministic and
stochastic automata operating within random environments. Tsetlin's automaton represents the first model of learning automata. Over the past decade, learning automata have found diverse applications in various fields. These include: (1) **Pattern Recognition**: Convergence of Tsetlin machine for XOR, identity, and not operators [15,16]; (2) **Neural Networks**: Convolutional Regression, the similarity between perceptrons and Tsetlin, and deep neural networks [17–20]; (3) **NLP**: Pattern recognition tasks using propositional logic and semantic representation of words [21,22]; (4) **Optimization Problems**: Particle swarm and multilevel optimization [23–26]; (5) **Graph Theory**: Partitioning problem [27,28]; (6) **Computer Networks**: Cognitive radio, load balancing, and wireless networks [29–31]; (7) **Social networks** : Influence maximization [32–35] (8) **Multi-agent Systems** [36–38].

Our study focuses on two fundamental categories of learning automata, namely Fixed Structure Learning Automaton (FSLA) and Variable Structure Learning Automaton (VSLA) [14,39–41]. The FSLA belongs to the category of state machines, while the VSLA updates its strategy based on a probability vector. For this research, we specifically chose the $L_{KN,K}$ as a representative of the fixed structure group, along with a specialized type of VSLA. This particular VSLA exhibits a dynamic set of available actions at each instance, where only a subset of actions is accessible. Termed Variable Action Set Learning Automaton (VASLA) [42], this type of LA serves as a key component in our novel LA framework.

* Corresponding author.
E-mail address: ali.nikhalat@aut.ac.ir (A. Nikhalat-Jahromi).

https://doi.org/10.1016/j.asoc.2024.112416
Received 26 July 2023; Received in revised form 18 September 2024; Accepted 20 October 2024
Available online 6 November 2024

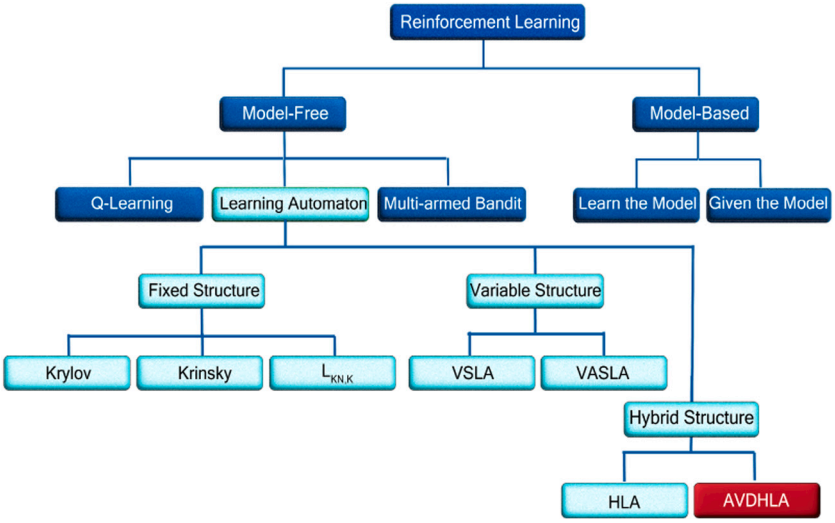


Fig. 1. The taxonomy of reinforcement learning algorithms based on the model.

The $L_{KN,K}$ learning automata are generally used in online learning decision-making contexts, making offline depth adjustments impractical [14,43]. Consequently, an online mechanism is crucial for adapting the depth of $L_{KN,K}$ based on specific problem requirements. This allows for incremental, decremental, or unchanged depth adjustments for each action. Such a mechanism will enable the $L_{KN,K}$ to efficiently explore and exploit the environment [44–46]. Lower depth values can lead to more exploration and frequent action switching, potentially incurring higher costs. In contrast, higher depth values may hinder effective action switching, increasing the exploitation of sub-optimal actions and reducing the learning automaton's efficacy. Thus, it is essential to adaptively adjust the depth of each action in the $L_{KN,K}$ to optimize its performance [14].

Furthermore, from the perspective of Automated Machine Learning (AutoML) [47–49], the $L_{KN,K}$ lacks internal mechanisms or tools to determine whether the selected depth for each action is appropriate or requires adjustment based on environmental conditions [14,43,50]. Therefore, an equal number of learning agents as the number of actions in the $L_{KN,K}$ should be integrated with it. These auxiliary agents can aid the $L_{KN,K}$ in modifying the depth of each action based on rewards received from the environment.

Given the exploration–exploitation dilemma and the principles of AutoML, selecting an appropriate depth for the $L_{KN,K}$ remains a significant challenge without current solutions. Our study introduces a novel solution to this critical problem. We propose the Asymmetric Variable Depth Hybrid Learning Automaton (AVDHLA), a hybrid learning model that combines fixed structure and variable action set learning automata. The AVDHLA effectively addresses the trade-off between exploration and exploitation by learning the appropriate depth, providing a solution to this challenge.

This work presents several key contributions compared to existing literature:

1. We pinpoint a crucial parameter, denoted as N, which serves as the depth parameter within the $L_{KN,K}$. This parameter presents a challenge in effectively balancing the exploration and exploitation capabilities of the $L_{KN,K}$.
2. We introduce a novel class of the $L_{KN,K}$ that possesses the ability to learn its depth in a fully self-adaptive manner. The proposed model determines the appropriate depth by utilizing the VASLA to learn from the performance of the $L_{KN,K}$ at various depths. Actually, each action's depth is controlled asymmetrically by its dedicated VASLA.
3. We complement our results with extensive computer simulations to evaluate the performance of the proposed learning model in both stationary and non-stationary environments, considering the total number of rewards and action switching. The results demonstrate the efficiency and improvement compared to the FSLA and the VSLA.
4. We examine the effectiveness of the learning model in a trendy practical application: Bitcoin [51], a decentralized cryptocurrency. The main drawback of the Bitcoin consensus mechanism is that a type of attack called selfish mining might threaten its decentralization and fairness capabilities. This study uses the AVDHLA to defend against the selfish mining attack [52,53]. Then, we compare our defense mechanism with the well-known defense called tie-breaking. Additionally, the results are compared with the defense mechanism that uses the $L_{KN,K}$. Analysis of the results shows the superiority of the proposed model in practical applications.
5. We showcase the practical versatility of the AVDHLA through the development of an innovative recommendation system that leverages its adaptive learning capabilities. This approach dynamically optimizes decision-making, leading to substantial improvements in key performance metrics such as CTR and Precision.

The remaining sections of this paper are organized as follows: Section 2 reviews previous work in this field. Section 3 provides the necessary background on learning automaton concepts. Section 4 formally states the problem. Section 5 elaborates on the proposed learning automaton. Section 6 presents the experimental results. Section 7 examines the application of the proposed model in blockchain security. Section 8 explores its application in recommendation systems. Section 9 discusses the strengths and weaknesses of the proposed model. Finally, Section 10 concludes the paper.

2. Related work

Reinforcement learning (RL), a pivotal paradigm in machine learning, can be categorized into two main types [1]: model-based [54,55] and model-free [56–58] approaches (Fig. 1). In model-based RL, agents construct an internal model of the environment to plan and optimize actions. In contrast, model-free RL involves agents learning directly from interactions with the environment, refining actions through trial and error. The choice between these approaches depends on the task characteristics, with model-based methods suited for environments where a reliable model is available, and model-free methods excelling

in complex and uncertain environments. In this paper, we have chosen learning automaton from the model-free category.

The first class of the learning automaton family is the fixed structure. The domain of the fixed structure learning automaton [59,60] is expansive, encompassing various branches such as $L_{KN,K}$ (Tsetlin) [43,61,62], Krinsky [63,64], and Krylov [65]. Each learning automaton is dedicated to specific decision-making paradigms, aligning with diverse cognitive abilities observed in human behavior. While the $L_{KN,K}$ incorporates reward and penalties, other models explore cognitive aspects such as impulsivity and greed. Recognizing the significance of the $L_{KN,K}$ within the learning automaton family and the limited attention it has received, we directed our focus toward this specific type.

The $L_{KN,K}$ [66] stands as a pioneering milestone in the domain of model-free reinforcement learning, representing a groundbreaking approach inspired by human cognitive processes. Originally rooted in psychology [67], this automaton encapsulates the intricate nature of human decision-making and calculation. Over its illustrious half-century existence, the $L_{KN,K}$ automaton [43,68] has demonstrated its versatility and applicability across a wide spectrum of fields [14]. Notably, its recent integration with Neural Networks [21,22,50,61,62,69–73] has gathered remarkable attention from researchers, propelling it to new heights of popularity in cutting-edge AI research.

Despite the $L_{KN,K}$'s effectiveness in stochastic environments, where quick trial-and-error solutions are essential, it encounters two fundamental challenges that impact its performance across various applications [14]: (1) The clockwise policy for selecting the next action; (2) The fixed depth value.

The Variable Structure Learning Automaton [14], belonging to the second class of learning automata, includes various types, with the VSLA and the VASLA being the most prominent branches. In the VSLA, interaction with the surrounding environment updates a probability vector. The VASLA is similar to the VSLA in its use of a probability vector; however, in certain situations, not all actions are available. Consequently, the VASLA selects the next action based on the set of actions available at each stage.

It is noteworthy that both the VSLA and the VASLA have a wide range of applications, including: (1) Social Networks [74,75]; (2) Fog Computing [76,77]; (3) Recommender Systems [78]; (4) Computer Networks [79,80]; (5) Optimizations [36,81,82].

Gholami et al. [83] introduced a significant modification to the $L_{KN,K}$ [83] by combining the fixed structure and the variable structure families, thereby establishing the third class and the state-of-the-art family of learning automata. This alteration specifically targeted the clockwise policy for selecting the next action.

Nevertheless, the challenge of dynamically adapting the depths of each action remains unresolved, limiting the balance between exploration and exploitation. To address this gap, our paper introduces a novel technique, expounded upon in the following sections, aimed at effectively addressing the issue of learning appropriate depth.

3. Preliminaries

A learning automaton [14,39] constitutes a model-free approach to reinforcement learning, engaging with an environment to acquire knowledge. In each time step, denoted as t, the learner makes a selection of an action, denoted as $\alpha(t)$, from the available action set $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$. Subsequently, the environment evaluates the chosen action, emitting a reinforcement signal denoted as $\beta(t)$ from the set $\beta = \{\beta_1, \beta_2, \ldots, \beta_m\}$. This process is illustrated in Fig. 2.

In the rest of this section, we will go deeper into an in-depth introduction of the FSLA, the VSLA, and the VASLA, which stand as key members within the learning automaton family.

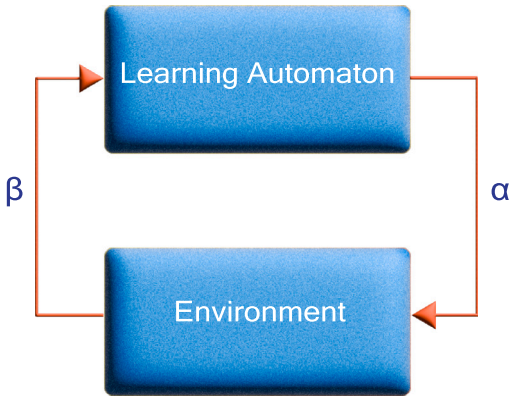


Fig. 2. The interaction of LA and the surrounding environment.

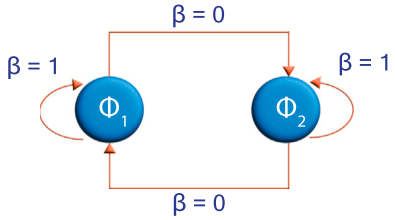


Fig. 3. The $L_{2,2}$ learning automaton.

3.1. Fixed Structure Learning Automaton (FSLA)

The fixed structure learning automaton [14,39] has a wide variety such as Tsetlin, $G_{2N,2}$, Ponomarev, Krylov, and so on. Since the $L_{KN,K}$ learning automaton is the most famous form of the fixed structure family, we concentrate on it in our study. The first part of this subsection is dedicated to the $L_{2N,2}$, which is a two-state $L_{KN,K}$. Afterward, the $L_{2N,2}$ is expanded to obtain the $L_{KN,K}$ as a general form of this family.

The $L_{2,2}$ or two-state learning automaton is the simplest form of the $L_{KN,K}$ family with ϕ_1, ϕ_2 states, and α_1, α_2 actions. The reinforcement signal of the environment comes from $\{0, 1\}$ set. The learning automaton will stay in the same state if the corresponding response is favorable ($\beta = 1$). On the other hand, if the corresponding response is not favorable ($\beta = 0$), the state of the learning automaton will switch. Fig. 3 depicts the structure of $L_{2,2}$.

Since the depth of $L_{2,2}$ equals 1, there is an action switching upon receiving an unfavorable response. To tackle this problem, the number of states increases from 1 to N for each action. These changes in states will convert the $L_{2,2}$ to the $L_{2N,2}$. The $L_{2N,2}$ has $2 \times N$ states and 2 actions. Such an automaton can incorporate the system's past behavior in its decision rule for choosing an appropriate action. In contrast with the $L_{2,2}$, which switches from one action to the other action on receiving an unfavorable response, $L_{2N,2}$ preserves the number of successes and failures received for each action. When the number of failures goes beyond the number of successes by one, the automaton will switch to the other action. The $L_{2N,2}$ learning automaton with $N = 2$ is shown in Fig. 4.

Till now, what has been described is an automaton with only two actions. However, this idea can be generalized to cases where the automaton can perform K actions. The major difference between a learning automaton with K actions and a learning automaton with two actions relates to switching from one action to the next. This learning automaton is called $L_{KN,K}$. In the following paragraph, this learning automaton is explained comprehensively.

The $L_{KN,K}$ has K actions $\alpha_1, \alpha_2, \ldots, \alpha_K$ and KN states $\phi_{(1,1)}, \phi_{(1,2)}, \ldots, \phi_{(1,N)}, \ldots, \phi_{(K,N)}$. Each state consists of an ordered pair (i, j) $(1 \leq i \leq K, 1 \leq j \leq N)$. i indicates the action number, and j shows the state

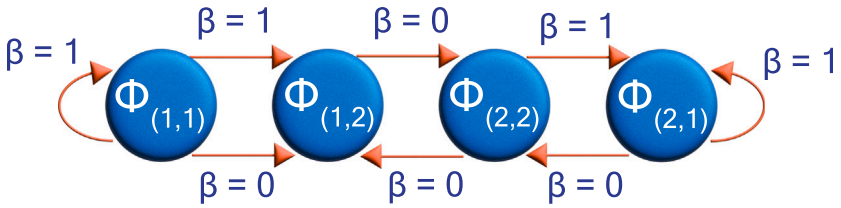


Fig. 4. The $L_{2N,2}$ learning automaton with $N = 2$.

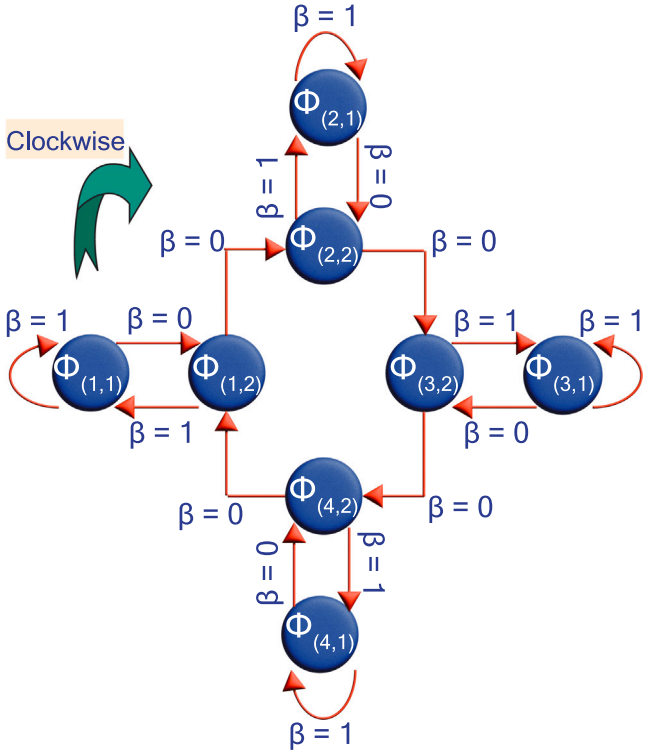


Fig. 5. The $L_{KN,K}$ learning automaton with $K = 4$ and $N = 2$.

number. If the automaton is in a state $\phi_{(i,j)}$, it performs the action α_i. By receiving an unfavorable response, the state changes as follows:

$$\begin{cases} \phi_{(i,j)} \rightarrow \phi_{(i,j+1)} & (1 \leq j \leq N-1) \\ \phi_{(i,j)} \rightarrow \phi_{(i+1,j)} & (j = N) \end{cases} \tag{1}$$

Furthermore, if the automaton receives a favorable response, the state will change as follows:

$$\begin{cases} \phi_{(i,j)} \rightarrow \phi_{(i,j-1)} & (2 \leq j \leq N) \\ \phi_{(i,1)} \rightarrow \phi_{(i,1)} & (j = 1) \end{cases} \tag{2}$$

Choosing the next action in this automaton is in a clockwise manner ($\phi_{(i,j)} \rightarrow \phi_{(i+1,j)}$). The state transition graph of the $L_{KN,K}$ with $K = 4$ and $N = 2$ is shown in Fig. 5.

For example, in Fig. 5, suppose that the learning automaton starts in state $\phi_{(1,2)}$. Upon receiving a favorable response ($\beta = 1$), the learning automaton moves towards depth, so the state $\phi_{(1,2)}$ transitions to $\phi_{(1,1)}$. If the learning automaton receives an unfavorable response ($\beta = 0$), it should switch its action to state $\phi_{(2,2)}$. Choosing the next action is done in a clockwise manner, so the learning automaton will choose action α_2.

3.2. Variable Structure Learning Automaton (VSLA)

More advanced models can offer additional flexibility, considering stochastic systems in which the state transitions or action probabilities are updated at every instance using a reinforcement scheme. Such a learning automaton is called a variable structure [14,39].

The variable structure can be defined mathematically by a quadruple $\langle \alpha, \beta, P, T \rangle$, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ denotes the finite action set from which the learning automaton can select the intended action, $\beta = \{\beta_1, \beta_2, \ldots, \beta_k\}$ denotes the set of inputs (reinforcement signals) to the learning automaton, $P = \{p_1, p_2, \ldots, p_r\}$ denotes the action probability vector, such that p_i is the probability of choosing the α_i action, and T denotes the learning algorithm that is used to update the action probability vector in terms of the environment's response at time t, i.e., $p(t+1) = T[\alpha(t), \beta(t), p(t)]$.

The VSLA performs its chosen action on the environment at time t. The probability vector contained within the learning automaton will be updated by receiving the response. If the chosen action is rewarded by the environment ($\beta = 1$):

$$p_j(t+1) = \begin{cases} p_j(t) + \lambda_1(1 - p_j(t)) & \text{if } j = i \\ (1 - \lambda_1)p_j(t) & \forall\, j \neq i \end{cases} \tag{3}$$

Vice versa, if the chosen action is punished by the environment ($\beta = 0$):

$$p_j(t+1) = \begin{cases} (1-\lambda_2)p_j(t) & \text{if } j = i \\ \frac{\lambda_2}{r-1} + (1-\lambda_2)p_j(t) & \forall\, j \neq i \end{cases} \quad (4)$$

In the above Eqs. (3), (4), r is the number of actions that can be chosen by the automaton. λ_1 and λ_2 indicate the reward and penalty parameters that determine the amount of increase and decrease of the action probabilities.

λ_1 and λ_2 can have different values. Based on these values, the updating schema can be categorized as follows:

- $\boldsymbol{L_{R-P}}$: This updating scheme, which is called "linear reward-penalty", comes from the equality of the reward and penalty parameters ($\lambda_1 = \lambda_2$). When both are the same, the probability vector of the learning automaton increases or decreases at a monotonic rate.
- $\boldsymbol{L_{R-\epsilon P}}$: This updating scheme, which is called "linear reward-ϵ penalty", leads to a much greater value of the reward parameter in relation to the penalty parameter ($\lambda_1 \gg \lambda_2$).
- $\boldsymbol{L_{R-I}}$: When there is no penalty in an updating scheme ($0 < \lambda_1 < 1, \lambda_2 = 0$), this updating scheme is called "linear reward-inaction". The probability vector of the learning automaton will not change upon receiving an unfavorable response from the environment.
- $\boldsymbol{L_{P-I}}$: If the conducted probability vector in the learning automaton does not change by receiving the favorable action, this updating scheme is called "linear penalty-inaction" ($\lambda_1 = 0, 0 < \lambda_2 < 1$).
- **Pure Chance**: An updating scheme in which there is no penalty and reward parameter ($\lambda_1 = \lambda_2 = 0$) is called "Pure Chance". In this updating scheme, the probability vector of the automaton will not change in any conditions.

3.3. Variable Action Set Learning Automaton (VASLA)

Under some circumstances, the number of available actions of the learning automaton varies at each time step. To overcome this constraint, a subset of the variable structure learning automaton called the variable action set learning automaton [42] is defined. Like the variable structure, this automaton can be formulated by a quadruple $\langle \alpha, \beta, P, T \rangle$, where α is the finite action set, β indicates a set of input to the learning automaton (reinforcement signal), P denotes the action probability vector, and T is the learning algorithm.

Such a learning automaton has a finite set of r actions denoting $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$. At each stage t, the action subset $\hat{\alpha} \subseteq \alpha$ is available for the learning automaton to choose from. Both action selection and updating the action probability vector in this learning automaton are described below.

Let $K(t) = \sum_{\alpha_i \in \hat{\alpha}(t)} P_i(t)$ present the sum of probabilities of the available actions in subset $\hat{\alpha}$. Before choosing an action, the available action probability vector is scaled using the following equation:

$$\hat{P}_i(t) = \frac{p_i(t)}{K(t)} \quad \forall \alpha_i \quad (5)$$

The crucial factor affecting the performance of the variable action set learning automaton is the learning algorithm for updating the action probabilities. Let α_i be the action chosen at time step t as a sample realization from distribution $p(t)$. Let λ_1 and λ_2 be the reward and penalty parameters, and r denotes the number of available actions. If the learning automaton chooses its intended action ($i = j$), the probability vector will update using the following equation:

$$p_i(t+1) = p_i(t) + \lambda_1\beta(1 - p_i(t)) - \lambda_2(1-\beta)p_i(t) \quad (6)$$

Conversely, the probability vector for the other actions ($i \neq j$) that are not chosen will update due to the next equation:

$$p_j(t+1) = p_j(t) - \lambda_1\beta p_j(t) + \lambda_2(1-\beta)[\frac{1}{r-1} - p_j(t)] \quad (7)$$

4. Problem formulation

In the classical $L_{KN,K}$ with K actions, each action i has N states denoted as $\phi_{(i,1)}, \phi_{(i,2)}, \ldots, \phi_{(i,N)}$ ($\phi_{(i,1)}$ is the last state or depth state, and $\phi_{(i,N)}$ is the first state or edge state). At time step t, the automaton selects action i based on the state $\phi_{(i,j)}$ ($1 \leq i \leq K$ and $1 \leq j \leq N$). The environment responds to the automaton at time step $t+1$. If the automaton is rewarded, $\phi_{(i,j)}$ will transition to $\phi_{(i,j-1)}$, except for $\phi_{(i,j=1)}$, which is a depth state. Conversely, if the environment penalizes the automaton, $\phi_{(i,j)}$ will transition to $\phi_{(i,j+1)}$, except for the edge state $\phi_{(i,j=N)}$. In such a situation, the automaton will change its action to $i+1$ based on the clockwise policy, and the next state will be $\phi_{(i+1,j=N)}$.

This model faces two critical challenges. Firstly, when the automaton intends to switch its action on the edge state $\phi_{(i,j=N)}$, it adheres to a clockwise policy for selecting the next action. Secondly, the depth N remains fixed. However, this assumption is inaccurate, especially in a non-stationary environment where the automaton may operate. To address these issues, we propose leveraging the capabilities of other automata to assist the $L_{KN,K}$ in making decisions for both the next action and its depth.

To address the clockwise policy challenge, Gholami et al. [83] employed a VSLA as a decision-maker to determine the next action. In their approach, when the system is in the state $\phi_{(i,j=N)}$ and the $L_{KN,K}$ receives a penalty, the VSLA activates to select the action $i+1$ based on its historical experiences. This collaboration between the VASLA and the $L_{KN,K}$ effectively resolves the clockwise policy issue in the $L_{KN,K}$.

To address the unsolved fixed-depth challenge, we propose the AVDHLA method. In this novel automaton, we employ K VASLA entities, denoted as $VASLA_1, VASLA_2, ..., VASLA_K$ to dynamically control the depth of each action. When an automaton performs action i, the corresponding $VASLA_i$ observes the number of transitions to the depth state $\phi_{(i,j=1)}$. When the automaton is on the edge state $\phi_{(i,j=N)}$ and intends to switch its action to $i+1$, the $VASLA_i$ selects an action from the set {'$Grow$','$Stop$','$Shrink$'}. Opting for the 'Grow' action increases the number of states for the i^{th} action, resulting in $\phi_{(i,1)}, \ldots, \phi_{(i,N)}, \phi_{(i,N+1)}$. Conversely, selecting the 'Shrink' action decreases the number of states, leading to $\phi_{(i,1)}, \ldots, \phi_{(i,N-2)}, \phi_{(i,N-1)}$. Choosing the 'Stop' action leaves the state unchanged. Over an extended duration, this collaborative interaction between the VASLAs and the $L_{KN,K}$ ensures the appropriate depth selection for each action.

5. Proposed algorithm : AVDHLA

In this section, a novel hybrid learning automaton will be introduced comprehensively. This new model of the automaton is called the "Asymmetric Variable Depth Hybrid Learning Automaton" and is abbreviated as "AVDHLA". AVDHLA consists of the $L_{KN,K}$ model of the fixed structure learning automaton and some variable action set learning automata. The distinctive feature of the proposed method lies in its approach to determining depth. Notably, there is no initial need to predefine the depth. Instead, the novel learning model autonomously establishes the appropriate depth based on the characteristics of the given environment. The next subsections will discuss the main algorithm, input parameters, and major units architecture. An illustrative example of the proposed model is provided in Appendix A. To have a better understanding of the proposed model, an architecture is depicted in Fig. 6.

5.1. AVDHLA's learning algorithm

AVDHLA, or Asymmetric Variable Depth Hybrid Learning Automaton, is the synthesis of two primary learning automaton models: 1- The fixed structure learning automaton which is used as the base learning model to interact with the outer environment, and 2- A number of variable action set learning automata. The number of VASLAs depends on the number of allowed actions.

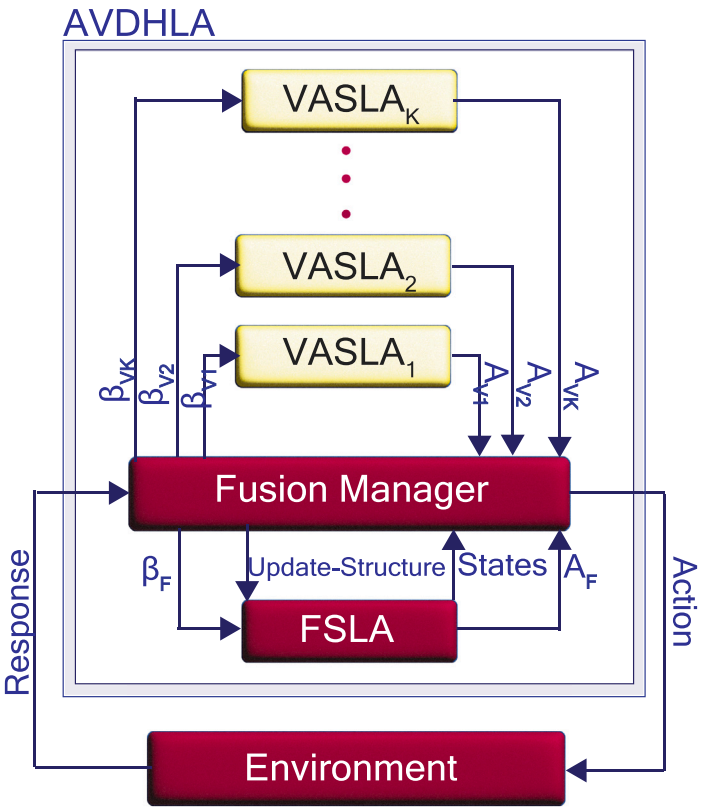


Fig. 6. The architecture of the AVDHLA.

The fixed structure learning automaton will perform exactly the same as what is described in Section 3.1. Changes of states by receiving the reinforcement signal from the outside environment in this automaton follow Eq. (1) (for unfavorable response) and Eq. (2) (for favorable response).

To clarify the environment setup, we designed an environment where the proposed automaton receives reinforcement signals from a finite set consisting of values 0 and 1.

In the proposed automaton, consider the fixed structure learning automaton approaches the edge state (the state with the highest number in the i^{th} action). The next punishment causes the FSLA to change the action. The conducted VASLA in the unfavorable action will activate to select the new depth for itself. This way of selecting helps the action to have a proper depth for the next times without affecting other action depth.

Choosing the new depth relies on the action-selection of the conducted variable action set in the i^{th} action among the predefined options:

1. **Grow**: If the $VASLA_i$ chooses this action, the depth of i^{th} action in the FSLA will increase by one.
2. **Shrink**: If the $VASLA_i$ chooses this action, the depth of i^{th} action in the FSLA will decrease by one.
3. **Stop**: If the $VASLA_i$ chooses this action, the depth of i^{th} action in the FSLA will remain the same as before.

Another point about the conducted VASLAs is the initial probability vector. Since there is no previous learning and there exist three actions, so each item in the probability vector is $\frac{1}{3}$ equally. In this situation, the VASLA chooses its action totally random. The next instances help the automaton to improve itself to set the proper depth.

Table 1
Table of notations.

Notation	Description
K	The number of allowed actions
N_i	The initial depth of i^{th} action
λ_1	The reward rate of inner VASLAs
λ_2	The penalty rate of inner VASLAs
$\phi_{(i,j)}$	The j^{th} state of the i^{th} action
α	The performing action of AVDHLA within the surrounding environment
β	The values of the reinforcement signal from the surrounding environment
A_{v_i}	The chosen action of i^{th} VASLA
β_{v_i}	The received response of i^{th} VASLA by preforming action A_{v_i}

5.2. AVDHLA's input parameters

The proposed model as a black box can be created with the constructor $AVDHLA(K, N_1, \ldots, N_i, \ldots, N_K, \lambda_1, \lambda_2)$. To make it simpler, it can be formulated as $AVDHLA(K, N, \lambda_1, \lambda_2)$.

The constructor is composed of these parameters: K is the number of allowed actions, N is the initial depth, λ_1 is the reward parameter, and λ_2 is the penalty parameter. It should be mentioned that if the extended constructor is used, N_i denotes the initial depth for the action number i ($1 \leq i \leq K$). Additionally, a table of notations (Table 1) has been included to simplify the comprehension of parameters.

5.3. AVDHLA's units architecture

As we can see in the architecture shown in Fig. 6, three major units construct the proposed model:

- **FSLA**: This unit implements the skeleton of the proposed learning automaton. This skeleton is based on the $L_{KN,K}$. Input parameters of the FSLA are: K, which denotes the number of actions, and N, which denotes the initial depth. In addition to two common predefined functions of the FSLA, namely *action_selection* (for selecting the next action) and $update(\beta)$ (for updating the reinforcement signal), three new functions are defined in this study:

    1. ***is_depth_transition***: This function examines the state transition to the depth of the automaton. If the automaton gets a reward and the next transition is the depth transition, this function returns true; otherwise, it will return false.
    2. ***is_action_switching***: This function checks for action switching. If the automaton receives a penalty from the surrounding environment and needs to change its action, this function returns true; otherwise, it returns false.
    3. ***update_depth(new_depth_length)***: This function can update the depth of all actions in the FSLA.

- $\boldsymbol{VASLA_i}$: This unit implements a variable action set learning automaton for the i^{th} action. This automaton has three actions ('Grow', 'Shrink', and 'Stop') for updating the depth of the i^{th} action. Each time the i^{th} automaton is activated, it performs action A_{v_i}, and β_{v_i} will be the received response. It is equipped with *action_selection* and $update(\beta_{v_i})$ functions. In the architecture of Fig. 6, K of these automata are shown ($1 \leq i \leq K$).
- **Fusion Manager**: This unit handles the connection of the FSLA and the VASLAs. As a matter of fact, this unit executes *action_selection* and $update(\beta)$ functions whenever they are needed.

5.3.1. Action-selection process

To manage the action-selection process of the FSLA and the corresponding VASLA of each FSLA's action together, a new function should be defined. Algorithm 1 shows this function. Two vectors are defined: one for the VASLA, in which the status of each VASLA is maintained, and another for the depth vector, in which the depth of each FSLA's action is shown at any moment.

In the *action_selection* function of i^{th} VASLA, one of two situations can occur:

1. If the automaton stays in the edge state and receives a penalty, the *action_switching* function will return a True value. In this condition, the proposed automaton should decide on the next depth of i^{th} action from one of these options: 'Grow', 'Shrink', and 'Stop'. Now, the automaton is responsible for its action-selection, so the chosen depth should remain the same until the next change in action.
2. Otherwise, if the automaton is not in the edge state, the $VASLA_i$ is dormant. In this case, the FSLA is responsible for choosing the next action.

5.3.2. Updating process

Each activated VASLA should update its probability vector upon action-selection, and when the FSLA decides to change its action, the probability vector should also be updated. The updating process is performed through Algorithm 2.

In the Algorithm 2, the notation β is used for the received reinforcement signal from the environment. $\beta = 0$ denotes an unfavorable action, while $\beta = 1$ denotes a favorable action. In addition to the β notation, there are new variables defined:

1. ***depth_counter***: If the automaton is rewarded for taking an action, this variable will count the number of transitions to the depth state (state with the lowest number). Furthermore, it will reset if the automaton changes its action.

Algorithm 1 action_selection()

Notation: $FSLA$ denotes the $L_{KN,K}$ type of fixed structure learning automaton,
$VASLA_i$ denotes the i^{th} variable action set learning automaton in the VASLA vector with 3 actions ($'Grow'$, $'Shrink'$, $'Stop'$),
N_i denotes the depth of FSLA's i^{th} action in the depth vector,
L denotes the last action made by the FSLA,
V_i denotes the last action made by the i^{th} VASLA

```
Begin
  if FSLA.is_action_switching() = True: then
    if N_L=1: then
      V_L = VASLA_L.action_selection(['Grow', 'Stop'])
    else
      V_L = VASLA_L.action_selection(['Grow','Stop','Shrink'])
    end if
    switch (V_L)
    case 'Grow':
      N_L = N_L + 1
      FSLA.update_structure(L, N_L)
    case 'Shrink':
      N_L = N_L - 1
      FSLA.update_structure(L, N_L)
    case 'Stop':
      /*Do nothing about structure*/
    end switch
    L = FSLA.action_selection()
    return L
  else
    L = FSLA.action_selection()
    return L
  end if
End
```

2. ***transition_counter***: This variable will count the number of transitions in one action. If the action changes, it will reset.
3. $\boldsymbol{\beta_{v_i}}$: When the automaton wants to change its action, this variable will be used to update the probability vector of the conducted $VASLA_i$(related to the i^{th} action). The following equation shows how to calculate it.

$$\beta_{v_i} = \frac{depth\_counter_i}{transition\_counter_i} \tag{8}$$

Based on the defined variables, Algorithm 2 works as follows:

- If the AVDHLA is rewarded ($\beta = 1$), one of these two occurrences may happen whether there is a depth transition or not:
    - If the depth transition happens (the FSLA may reach the depth state or stay in it), the depth_counter and the transition_counter variables will increase by one.
    - Otherwise, the depth transition does not occur, therefore the FSLA just receives the reward and goes to the inner state. The transition_counter variable increases due to performing a transition, while the depth_counter variable does not alter. As a result, the probability of being in the depth will decrease, and the conducted VASLA is prone to be penalized.
- Or else, the AVDHLA is penalized ($\beta = 0$). This causes the FSLA to be penalized as well. In this situation, the FSLA moves towards the edge state. The transition_counter increases, and the depth_counter stays the same as before. Consequently, the probability of being in the depth decreases, and the corresponding VASLA may be penalized.

6. Evaluation

This section is designed for the evaluation of the AVDHLA. To evaluate the correctness and efficiency of the proposed learning automaton, we first introduce the environments. Then, metrics for the evaluation are presented. Finally, various experiments have been designed to assess the new learning automaton and compare it with other kinds of learning automata.

6.1. Environment

The surrounding environment can be formulated as a triple $\langle \alpha, \beta, C \rangle$ in which [14]:

1. $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ denotes a finite set of actions received by the learning automaton.
2. $\beta = \{\beta_1, \beta_2\}$ denotes a binary set representing the responses from the environment to the learning automaton. For mathematical convenience, β_1 and β_2 are set to 0 and 1, respectively.
3. $C = \{c_1, c_2, \ldots, c_r\}$ denotes the penalty probability in which the element c_i may characterize the environment. c_i maps to the action α_i.

The input α_t is applied to the environment at the discrete time t. The output $\beta(t)$ of the environment can be one of the following:

- $\beta(t) = 1$: The environment responds positively to the applied action and as a result, the automaton receives a reward.
- $\beta(t) = 0$: The environment responds negatively to the applied action and consequently, the automaton will be punished.

By the above definitions, the negative response from the environment can be summarized in the following mathematical equation:

$$Pr(\beta(t) = 0 | \alpha(t) = \alpha_i) = c_i (i = 1, 2, \ldots, r) \tag{9}$$

Algorithm 2 update(β)

Notation: $FSLA$ denotes the $L_{KN,K}$ type of fixed structure learning automaton,
$VASLA_i$ denotes the i^{th} variable action set learning automaton in the VASLA vector with 3 actions ($'Grow'$, $'Shrink'$, $'Stop'$),
L denotes the last action made by the FSLA,
$depth\_counter_i$ denotes counter for counting depth transition of the $i^{th} action$,
$transition\_counter_i$ denotes counter for counting transitions of the i^{th} action,
β_{v_i} denotes the reinforcement signal of the i^{th} VASLA

```
Begin
  if β = 1: then
    if FSLA.is_depth_transition() = True: then
      depth_counter_L = depth_counter_L + 1
    end if
    FSLA.update(1)
    transition_counter_L = transition_counter_L + 1
  else if β = 0: then
    if FSLA.is_action_switching() = True: then
      β_{v_L} = depth_counter_L/transition_counter_L
      VASLA_L.update(β_{v_L})
      depth_counter_L = 0
      transition_counter_L = 1
    end if
    transition_counter_L = transition_counter_L + 1
    FSLA.update(0)
  end if
End
```

Meanwhile, the positive response can be shown as the following equation:

$$Pr(\beta(t) = 1 | \alpha(t) = \alpha_i) = 1 - c_i (i = 1, 2, \ldots, r) \tag{10}$$

From an operational standpoint [14], the environment can be categorized as *stationary* or *non-stationary*. Furthermore, non-stationary environments can be divided into *Markovian switching* and *State-dependent* environments.

6.1.1. Stationary environment

Referring to the previous explanations in the above section, c_i denotes the penalty probability at time t. If this probability does not change over time, the environment can be considered to be a stationary environment [14,39].

6.1.2. Non-stationary environment

Contrary to the stationary environment, whenever c_i varies over time, the environment is non-stationary [14].

If a learning automaton with a fixed strategy is used in such an environment, it may become useless or perform with a lot of penalties. To tackle this problem, the automaton should have enough flexibility to track the changes in the environment.

Looking closely at non-stationary environments may lead us to the conclusion that $c_i(t)$ may be constant over an interval $[t, t + T - 1]$, and switch to a new value at time $t + T$.

Explicitly, each penalty set can map to a new environment $E_i \in \{E_1, E_2, \ldots, E_d\}$. This mapping leads us to the result that learning in a time-varying environment corresponds to learning in multiple random environments.

MSE. MSE or *Markovian Switching Environment* is an environment where each E_i $(1 \leq i \leq d)$ environment is itself a state of a Markov chain. If the chain is ergodic, an automaton that is connected to such an environment will be in each of the component environments with a fixed probability corresponding to the asymptotic probability distribution of the ergodic chain [14,39].

State-Dependent Environment. If the environment E_i $(1 \leq i \leq d)$ has a finite number of sub-environments, and the state of each environment alters at the timestep t, such a composite environment is referred to as a *State-Dependent Environment* [14,39].

In this environment, state changes may depend on the time step t, either explicitly or implicitly. Generally, this type of environment is characterized by the penalty probabilities c_i.

State-dependent environments have three well-known models (A, B, C). In this study, the B and C models are ignored. Hence, model A is explained in detail.

In the A model of such an environment, if the automaton performs action α_i at the time step t, two events will occur:

1. The c_i probability of the i^{th} action will increase.
2. The c_i probability of the other actions $(j \neq i)$ decreases.

These occurrences imply that the performed action becomes worse for the next stages, while other actions improve with time.

From a mathematical standpoint, this environment is described by the following equations:

$$\begin{cases} c_i(t) = c_i(t) + \zeta_i(t) & j = i \\ c_j(t) = c_j(t) - \psi_j(t) & j \neq i \end{cases} \tag{11}$$

In the above equation, $\zeta_i(t)$ and $\psi_j(t)$ $(i, j = 1, 2, \ldots, r)$ are both non-negative functions of the t time step. Generally, $\zeta_i(t)$ and $\psi_j(t)$ might be function of $c_i(t)$.

To make the model simpler, $\zeta_i(t)$ is considered to be a constant value without any dependencies on the time step t. The following equation is used for $\zeta_i(t)$:

$$\zeta_i(t) = \begin{cases} \zeta_i & c_i(t) + \zeta_i(t) \leq 1 \\ 1 - c_i(t) & o.w \end{cases} \tag{12}$$

And $\psi_j(t)$ is also considered to be constant. The relationship between $\psi_j(t)$ and time t is ignored. The simplified equation is presented below:

$$\psi_j(t) = \begin{cases} \psi_j & c_j(t) - \psi_j(t) \geq 0 \\ c_j(t) & o.w \end{cases} \tag{13}$$

6.2. Metrics

Metrics for the evaluation of the proposed learning automaton are introduced here. These metrics will be applied to show the efficiency and performance of the AVDHLA in the designed experiments presented in various environments.

In the following items, these metrics are defined [14,39,83]:

- **Total Number of Rewards (TNR)**: This evaluation metric shows the total number of rewards that the automaton has received from the environment ($\beta = 1$). At the t time step, if the automaton receives positive feedback from the environment,TNR will be increased by 1. Generally, higher values of TNR indicate better performance of the learning automaton. The cumulative relation of TNR over T iteration is shown in the following equation [83]:
$$TNR_T = \sum_{t \leq T} \beta_t \tag{14}$$
- **Total Number of Action Switching (TNAS)**: This metric counts the number of times the automaton changes its chosen action within a given environment. Tracking action switches is essential, as some learning automata incur significant costs when switching actions. TNAS_T represents the total number of action switches, denoted by s_t at the time step t, that the automaton makes over T iterations. The equation for TNAS_T is shown below [83]:
$$TNAS_T = \sum_{t \leq T} s_t \tag{15}$$
We should remark that the lower number of TNAS is better for the automaton.
- **Probability of Choosing the Favorable Action ($P(\alpha_i)$)**: If the automaton has a situation in which it can choose between more than one action and one of them is favorable, the probability of choosing the favorable action should be considered as an evaluation metric [83].
This parameter equals the total number of times that α_i is chosen divided by the total number of times that other actions have been chosen.
$$P(\alpha_i) = \frac{Number\ of\ times\ that\ \alpha_i\ is\ chosen}{Total\ number\ of\ action\ selection} \tag{16}$$
The higher values for $P(\alpha_i)$ show that the automaton is moving towards the best possible action. Hence, this parameter has an important role in the experiments.

6.3. Experimental results

We designed numerous experiments to test the proposed automaton in all possible aspects. These experiments were conducted in various environments which were introduced in Section 6.1.

For the first time in the literature, an automaton was tested in the *Markovian switching* and *State-dependent* environments. This novel approach will help researchers to see the strengths and weaknesses of the learning automaton in such environments.

Since three kinds of environments had been introduced, three major categories of experiments were designed. These categories are:

1. **Experiment1**: The main goal of this experiment is to evaluate the proposed automaton in an environment with fixed penalty probability (c_i).
2. **Experiment2**: This category of experiments is dedicated to testing the proposed automaton in the Markovian switching environments with different properties for each environment.
3. **Experiment3**: In this category of experiments, the proposed automaton is assessed in different kinds of state-dependent environments.

To conduct the aforementioned experiments, we developed a Python program to simulate each of the environments:

- In stationary environments, a random function generates a number according to the Uniform distribution. Subsequently, each action performed by the automaton undergoes evaluation based on a predefined threshold in the experiments. If the generated number is below the threshold, indicating an inappropriate action, the automaton is subjected to a penalty; conversely, if the number exceeds the threshold, signifying a suitable action, the automaton receives a reward.
- In Markovian environments, the designed setting comprises multiple stationary environments, each corresponding to a distinct state defined within the system. Transitions between states trigger changes in the environment, effectively influencing the dynamics of the Markovian system.
- In state-dependent environments, the reward and penalty thresholds for the environment dynamically change based on the chosen actions of the automaton. While the implementation of these environments resembles that of stationary environments, the key distinction lies in the adaptive nature of the threshold, which undergoes alterations in each time step.

It is important to highlight our utilization of the T-test for statistical analysis in this study. Each experiment was performed 50 times, and the average values for both rewards and action switching were calculated. Subsequently, the results of T-tests are presented in each table, reporting the p-value parameter. A p-value less than 0.05 is considered indicative of a significant difference between the two groups.

Our code is accessible on GitHub[1] to facilitate further research endeavors. Additionally, the experiments detailed in this subsection were conducted on an Intel Core i7 processor operating at a clock frequency of 2.5 GHz.

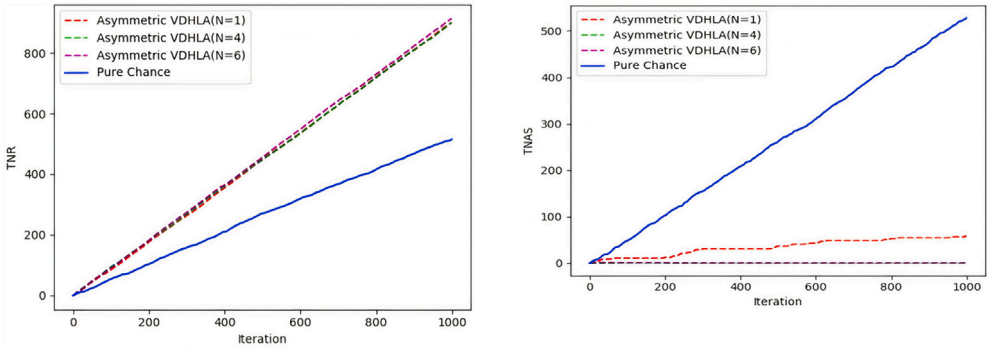
6.3.1. Experiment 1 - Stationary environment

This experiment is designed to evaluate the proposed learning automaton in stationary environments with fixed penalty probability. To have a better overview of the proposed automaton in stationary environments, Experiment 1 is divided into four sub-experiments.

- Experiment 1.1 investigates the learning capability of the automaton.
- Experiment 1.2 is designed to compare the proposed automaton with the FSLA.
- Experiment 1.3 studies the proposed automaton versus the VSLA.
- Experiment 1.4 aims to compare our proposed automaton with the HLA, a leading method in the $L_{KN,K}$ field.

Experiment 1.1. The learning capability of the proposed automaton should be considered before conducting any other experiments. To achieve this, Experiment 1.1 is designed to evaluate whether the proposed automaton performs better than the *Pure Chance Automaton (PCA)*, which chooses an action randomly at each time step.

[1] https://github.com/AliNikhalat/LearningAutomata.



(a) TNR of AVDHLA (b) TNAS of AVDHLA

Fig. 7. Experimental results of Experiment 1.1 with reward probability vector of (0.1, 0.9).

Table 2
Experimental results of Experiment 1.2 for the AVDHLA.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
AVDHLA	**7756.10 ± 346.36**	**364.44 ± 509.08**	**7793.68 ± 345.82**	**294.86 ± 516.73**	7957.08 ± 68.86	71.34 ± 104.80	**8006.46 ± 40.97**	**5.32 ± 5.35**
FSLA	4565.98 ± 86.08	5434.02 ± 86.08	7695.62 ± 62.09	455.36 ± 55.35	**7982.56 ± 43.01**	**32.18 ± 16.84**	7988.36 ± 41.47	5.74 ± 6.17
P-Value	$\mathbf{8.15 * 10^{-81}}$	$\mathbf{1.00 * 10^{-84}}$	**0.05**	**0.03**	**0.03**	**0.01**	**0.03**	0.71

To examine the learning capability of the AVDHLA, an environment is designed that considers the probability of being rewarded for corresponding actions. The probability of action 1 to get a reward from the environment is 0.1, and the probability of action 2 is 0.9.

For the sake of explanation, the first action (action α_1) has a probability of 0.1. This means that the automaton in this environment will be rewarded 10% of the time by choosing α_1. Meanwhile, choosing the second one (action α_2) brings the performed action of the automaton with a reward 90% of the time.

In such an environment, the automaton must incline towards the second action more than the first one to obtain a higher number of rewards and decrease its number of action switching.

This experiment is performed for the AVDHLA in 1000 iterations. Considered automaton has two actions, and the inner VASLAs are $L_{R-\varepsilon P}$ with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$. Also, they have $(1, 1)$, $(4, 4)$, and $(6, 6)$ initial depth. Fig. 7 shows the result from the TNR and TNAS aspects.

As these figures show, the AVDHLA outperforms the pure chance automaton with respect to TNR and TNAS. This leads us to the result that the AVDHLA is able to learn effectively. Furthermore, according to the results of this experiment, the AVDHLA can detect the action with a higher probability of reward very well. Also, the AVDHLA with a higher depth can adapt itself to the environment faster than the others with lower values.

The main reason for this faster adoption is the AVDHLA's ability to increase its depth for favorable actions. An AVDHLA with a higher depth has a better initial condition and, therefore, requires less effort to adjust its depth. As a result, it will obtain more rewards and reduce the number of action switches towards favorable actions.

Two other configurations are considered in Appendix B.1 to evaluate the learning capability of AVDHLA. The first one has probabilities closer to each other (0.3 for action α_1 and 0.7 for action α_2). The next one has the same reward probability for both actions (0.5 for action α_1 and 0.5 for action α_2).

Experiment 1.2. This experiment is conducted to compare the proposed learning automaton with a fixed structure learning automaton. The considered environment has just one favorable action with a probability of 0.8. Depending on the number of other actions, 0.2 will be divided among them. The following equation summarizes the configuration of the environment from the perspective of the probability of being rewarded.

$$P(\alpha_i) = \begin{cases} 0.8 & i = 1 \\ \frac{0.2}{K-1} & i = 2, \dots, K \end{cases} \tag{17}$$

The initial depth considered for the AVDHLA is 1, 3, 5, and 7 for configuration numbers 1 to 4, respectively. The number of iterations is 10000 because the AVDHLA needs more time to adapt to the environment. Additionally, the inner VASLAs are $L_{R-\varepsilon P}$ with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$. The results are shown in Table 2.

The results indicate that, in most scenarios, the AVDHLA outperforms the FSLA in both TNR and TNAS metrics. This suggests that the AVDHLA is less sensitive than the FSLA to the initial depth. This advantage is more apparent at lower depths, as the AVDHLA can autonomously increase its depth. When the FSLA's depth is increased, it can exploit more effectively toward the optimal action, resulting in more efficient reward acquisition and a reduction in the total number of action switches. Conversely, at lower depths, the FSLA tends to engage in excessive exploration, leading to a decrease in the number of rewards obtained.

As well as TNR and TNAS metrics, we consider the probability of choosing the favorable action. In this experiment, the favorable action is α_1 and the number of allowed actions is 9. The results can be seen in Fig. 8. By looking at this figure, we can see that the AVDHLA impressively finds its appropriate depth in terms of the favorable action. It means that AVDHLA is robust in terms of changing memory depth, which is a great benefit of this model.

We should note that additional experiments were conducted to provide a more comprehensive comparison between the AVDHLA and the FSLA. These experiments are detailed in Appendix B.2.

Experiment 1.3. An experiment needs to be designed to compare the AVDHLA with the VSLA. In order to do this, an environment where the probability of being rewarded for a specific action is 80% is considered. In such an environment, others have roughly 20% chance to receive a reward. Eq. (17) is applied to create this environment.

Besides, the number of allowed actions for an automaton to be in this environment is 4. Five configurations are used for the VSLAs to compare with the AVDHLA. Each configuration is tested with two

Table 3
The setup of Experiment 1.3.

	P	L_{R-I}		L_{P-I}		L_{R-P}		$L_{R-\epsilon P}$	
	Config1	Config2.a	Config2.b	Config3.a	Config3.b	Config4.a	Config4.b	Config5.a	Config5.b
K	5	5	5	5	5	5	5	5	5
N	5	5	5	5	5	5	5	5	5
λ_1	0	0.1	0.01	0	0	0.1	0.01	0.1	0.01
λ_2	0	0	0	0.1	0.01	0.1	0.01	0.01	0.001

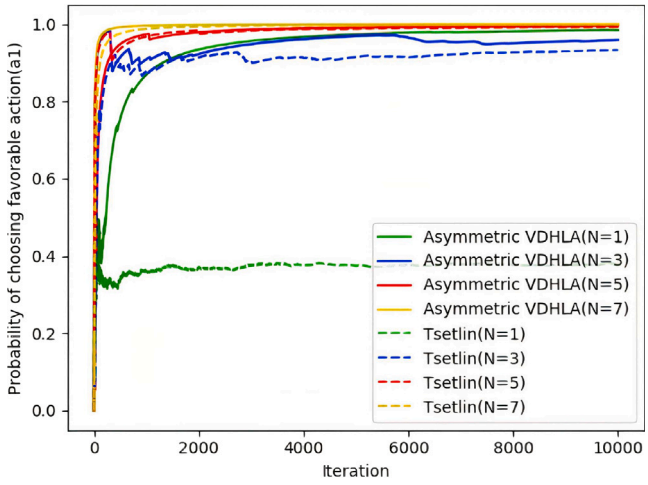


Fig. 8. Experimental results of Experiment 1.2 with respect to the probability of choosing the favorable action for the AVDHLA with $K = 9$.

values for λ_1 and λ_2 (denoted by "a" and "b"). These configurations are displayed in Table 3 from number 1 to 5.

The results of this experiment with respect to TNR and TNAS are shown in Table 4, considering the AVDHLA for 1000 iterations. These outstanding results show the power of the proposed automaton to collect more rewards with a lower number of action switching over the VSLA in all cases. It means that AVDHLA, unlike the VSLA, does not have any dependencies on updating schemes. Also, the lower number of action switching indicates that the proposed automaton learns to find the favorable action.

Moreover, in the probability of choosing the favorable action metric, the VSLA is inferior to the AVDHLA. This comes from our measuring of this metric in Fig. 9 for the AVDHLA. The main reason is that the AVDHLA inclines to the action α_1 swiftly, therefore, it is less likely to receive a penalty from the environment. Also, this part of the experiment leads us to the result that the probability of choosing the favorable action in the AVDHLA, in contrast to the VSLA, is independent of the updating scheme.

Examining all experiments in Fig. 9, it is evident that the pure chance VSLA exhibits the worst performance due to its lack of learning capabilities. Conversely, in the corresponding AVDHLA experiment, where action switching and depth changing occur, the AVDHLA demonstrates success. Moreover, the best performances are observed in L_{R-P} and $L_{R-\epsilon P}$, as they offer a superior understanding of the stationary environment by incorporating penalty rates.

Experiment 1.4. This experiment is conducted to compare the proposed learning automaton with the state-of-the-art method called HLA. The considered environment is similar to Experiment 1.2, except for the inner VASLA. We used L_{R-I} with a reward rate equal to 0.01. The results are shown in Table 5.

As is evident from the results, in most cases, the AVDHLA outperforms the HLA in terms of both TNR and TNAS. This is particularly noticeable at lower depths, especially in the first configuration. The difference arises because, unlike the HLA, the AVDHLA dynamically sets its depth based on its interaction with the surrounding environment. This will help achieve a balance between exploration and exploitation. Additionally, in all scenarios, the AVDHLA adjusts its depth to minimize TNAS.

6.3.2. Experiment 2 - Markovian switching environment

This experiment is conducted to study the performance of the proposed learning automaton in the complex Markovian switching environment with respect to TNR and TNAS. To reach this objective, a Markov chain with 4 states is considered. To have a better overview, Fig. 10 shows the desired environment. In such an environment, we considered the transition matrix (T) and the probability of being rewarded in each state denoted by the reward matrix (R), respectively.

$$T = \begin{pmatrix} 0.3 & 0.2 & 0.1 & 0.4 \\ 0.1 & 0.2 & 0.5 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.6 \\ 0.2 & 0.5 & 0.1 & 0.2 \end{pmatrix} \tag{18}$$

$$R = \begin{pmatrix} 0.9 & 0.1 & 0.3 & 0.7 & 0.1 \\ 0.1 & 0.9 & 0.7 & 0.6 & 0.2 \\ 0.3 & 0.7 & 0.5 & 0.5 & 0.3 \\ 0.9 & 0.9 & 0.9 & 0.4 & 0.6 \end{pmatrix} \tag{19}$$

In addition, the inner VASLAs are L_{R-I} with $\lambda_1 = 0.01$ and $\lambda_2 = 0$. The number of allowed actions is 5, with initial depths set to 1, 3, 5, and 7 for configurations 1 through 4, respectively. Each experiment is run for 10000 iterations.

This experiment is divided into two sub-experiments as follows:

- Experiment 2.1 is designed to compare the proposed learning automaton with the FSLA.
- Experiment 2.2 explores the performance of the proposed learning automaton against the HLA.

In addition to this experiment, an accurate analysis of the AVDHLA in Markovian switching environments is performed in Appendix C. In this Appendix, various conditions are considered to compare the proposed learning automaton with the FSLA in conjunction with TNR and TNAS.

Before conducting the experiments, it is advisable to transform the Markovian switching environment into a stationary one through a steady-state analysis of the Markov chain. This conversion yields reward probabilities of $[0.52, 0.68, 0.62, 0.52, 0.31]$ for actions 1 through 5, respectively.

Experiment 2.1. This experiment compares the performance of the AVDHLA with the FSLA. The results are presented in Table 6. Given the fairly random nature of the designed environment, the proposed learning automaton demonstrates its adaptability to the Markovian switching environment in terms of TNR and TNAS. It is worth noting that all actions in the designed environment have approximately the same probability of receiving a reward, so extremely high performance is not expected. Nonetheless, the AVDHLA outperforms the FSLA in reward acquisition and minimizing action switching, except in Configuration 4. The main reason for this exception lies in the AVDHLA's tendency to explore more to identify the appropriate depth, which slightly impacts its performance. In contrast, the FSLA maintains a fixed depth. Overall, these results confirm that the AVDHLA is a reliable choice for Markovian switching environments.

Table 4
Experimental results of Experiment 1.3 with respect to TNR and TNAS.

Model	Config1		Config2.a		Config2.b		Config3.a		Config3.b		Config4.a		Config4.b		Config5.a		Config5.b	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
AVDHLA	**793.78 ± 14.52**	**8.40 ± 10.95**	**791.82 ± 17.64**	**8.78 ± 15.12**	**791.86 ± 14.61**	**8.44 ± 10.39**	**793.94 ± 17.22**	**6.66 ± 9.96**	**793.98 ± 16.50**	**7.64 ± 8.93**	**792.08 ± 13.40**	**8.58 ± 9.21**	**796.16 ± 17.66**	**9.08 ± 18.31**	**790.2 ± 29.83**	**15.5 ± 35.29**	**796.44 ± 13.70**	**5.80 ± 6.09**
VSLA	197.84 ± 13.08	796.28 ± 13.35	783.1 ± 12.55	29.06 ± 8.93	227.88 ± 12.73	813.78 ± 12.58	220.94 ± 10.83	800.82 ± 11.05	223.16 ± 12.34	799.24 ± 12.83	449.94 ± 38.73	625.12 ± 39.83	403.64 ± 27.95	695.2 ± 25.64	758.5 ± 14.04	83.46 ± 16.58	613.26 ± 30.32	343.84 ± 36.85
P-Value	$\mathbf{1.60 * 10^{-132}}$	$\mathbf{1.09 * 10^{-149}}$	**0.005**	$\mathbf{1.70 * 10^{-12}}$	$\mathbf{1.41 * 10^{-130}}$	$\mathbf{5.10 * 10^{-153}}$	$\mathbf{3.38 * 10^{-129}}$	$\mathbf{2.39 * 10^{-156}}$	$\mathbf{1.74 * 10^{-128}}$	$\mathbf{4.21 * 10^{-154}}$	$\mathbf{5.63 * 10^{-78}}$	$\mathbf{9.70 * 10^{-103}}$	$\mathbf{1.15 * 10^{-92}}$	$\mathbf{2.79 * 10^{-118}}$	$\mathbf{1.55 * 10^{-9}}$	$\mathbf{2.26 * 10^{-21}}$	$\mathbf{5.26 * 10^{-61}}$	$\mathbf{2.48 * 10^{-81}}$

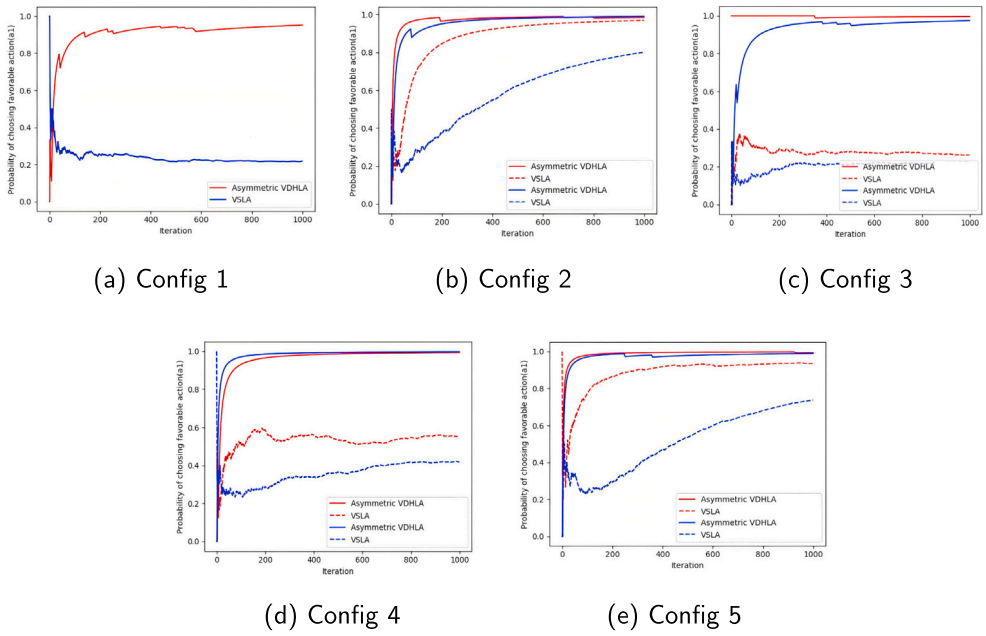


Fig. 9. Experimental results of Experiment 1.3 with respect to the probability of choosing the favorable action for the AVDHLA; Config a denoted by red and Config b denoted by blue.

Table 5
Results of Experiment 1.4: Performance comparison between the AVDHLA and the HLA.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
AVDHLA	**7880.54 ± 108.15**	**188.16 ± 125.35**	**7904.2 ± 83.55**	**141.82 ± 111.70**	7971.72 ± 47.18	**39.18 ± 48.62**	**7999.26 ± 39.65**	**5.82 ± 8.62**
HLA	7818.98 ± 46.00	2181.76 ± 45.94	7857.56 ± 42.63	287.32 ± 35.32	**7972.44 ± 43.36**	39.68 ± 17.68	7983.3 ± 40.13	9.3 ± 7.50
P-Value	**0.006**	$\mathbf{9.80 * 10^{-71}}$	**0.0007**	$\mathbf{8.21 * 10^{-14}}$	0.93	0.94	**0.05**	**0.03**

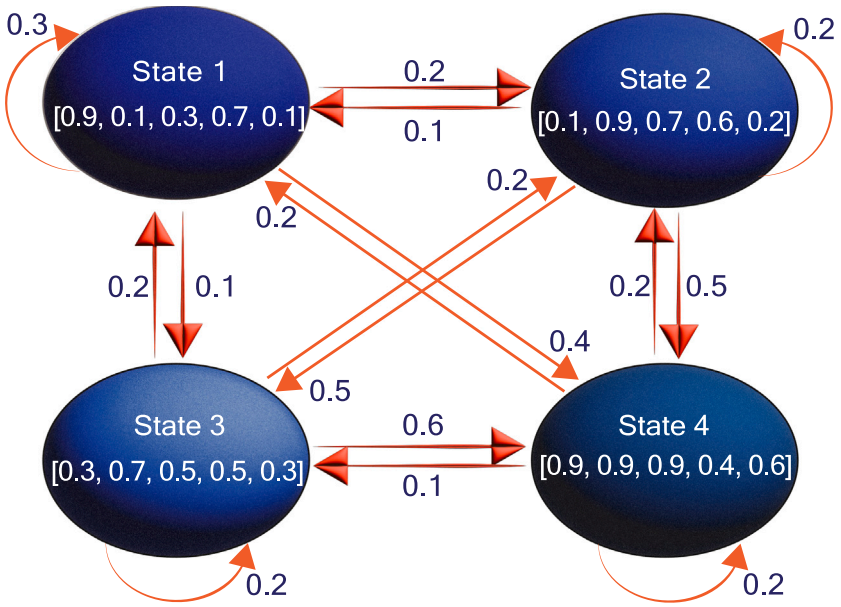


Fig. 10. The designed environment for the Markovian switching experiment.

Experiment 2.2. This experiment is conducted to compare the AVDHLA with the HLA, and the results are shown in Table 7. Despite the inherently random nature of the environment, the AVDHLA outperforms the HLA in most cases in terms of TNR and TNAS. This trend is particularly evident in lower depths, attributable to the flexibility of the AVDHLA in choosing an appropriate depth.

6.3.3. Experiment 3 - State-dependent environment

This experiment is devoted to studying the performance of the proposed learning automaton in a State-dependent environment with respect to TNR and TNAS. To achieve this, the experiment is divided into three parts:

Table 6
Experimental results of Experiment 2.1 with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
AVDHLA	**6498.06 ± 287.92**	**286.68 ± 150.47**	**6527.52 ± 281.73**	**219.02 ± 113.98**	**6658.42 ± 246.14**	**126.76 ± 83.84**	6677.28 ± 230.22	82.6 ± 67.75
FSLA	5825.84 ± 52.21	4174.16 ± 52.21	6250.84 ± 68.65	851.6 ± 43.43	6573 ± 87.74	227.68 ± 38.98	**6752.5 ± 112.61**	**63.28 ± 20.42**
P-Value	$\mathbf{3.06 * 10^{-29}}$	$\mathbf{4.06 * 10^{-123}}$	$\mathbf{1.46 * 10^{-9}}$	$\mathbf{1.23 * 10^{-58}}$	**0.02**	$\mathbf{1.47 * 10^{-11}}$	**0.04**	**0.05**

Table 7
Experimental results of Experiment 2.2 with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
AVDHLA	6536.68 ± 277.15	**236.48 ± 116.29**	**6580.0 ± 297.92**	**220.32 ± 121.36**	**6612.22 ± 223.25**	**150.04 ± 95.66**	6706.5 ± 218.30	**61.22 ± 38.72**
HLA	**6628.04 ± 110.51**	3372.66 ± 110.51	6489.6 ± 107.03	957.52 ± 87.19	6611.16 ± 114.50	323.24 ± 71.20	**6741.2 ± 121.71**	101.6 ± 28.82
P-Value	**0.03**	$\mathbf{1.03 * 10^{-113}}$	**0.04**	$\mathbf{1.16 * 10^{-56}}$	0.97	$\mathbf{5.27 * 10^{-17}}$	0.33	$\mathbf{6.34 * 10^{-8}}$

Table 8
Experimental results of Experiment 3.1 with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
Scenario 1 - ($\zeta = 0.0002, \psi = 0.00002$)								
AVDHLA	**788.68 ± 14.83**	**35.58 ± 22.86**	**794.98 ± 13.99**	**18.76 ± 16.42**	**802.02 ± 14.85**	3.76 ± 6.94	**803.36 ± 12.67**	**1.04 ± 1.21**
FSLA	696.62 ± 13.05	303.38 ± 13.05	788.56 ± 15.60	24.74 ± 9.38	796.86 ± 10.59	**3.36 ± 2.91**	797.46 ± 12.21	1.2 ± 1.44
P-Value	$\mathbf{2.08 * 10^{-54}}$	$\mathbf{3.40 * 10^{-86}}$	**0.03**	**0.02**	**0.05**	0.71	**0.02**	0.55
Scenario 2 - ($\zeta = 0.0005, \psi = 0.0005$)								
AVDHLA	**654.66 ± 13.78**	**81.86 ± 29.63**	**660.06 ± 16.31**	**67.74 ± 25.72**	**663.64 ± 12.74**	46.34 ± 22.31	**662.16 ± 14.82**	30.84 ± 15.87
FSLA	638.84 ± 15.20	361.16 ± 15.20	654.36 ± 12.67	83.56 ± 13.13	657.4 ± 13.81	**38.92 ± 14.05**	655.82 ± 13.44	**24.66 ± 10.45**
P-Value	$\mathbf{4.74 * 10^{-7}}$	$\mathbf{3.65 * 10^{-78}}$	**0.05**	**0.0002**	**0.02**	**0.05**	**0.028**	**0.025**
Scenario 3 - ($\zeta = 0.00002, \psi = 0.0002$)								
AVDHLA	**882.56 ± 12.05**	**15.92 ± 9.25**	**890.14 ± 10.66**	4.64 ± 6.36	**892.18 ± 8.27**	0.84 ± 1.20	**891.6 ± 9.43**	0.96 ± 0.93
FSLA	825.18 ± 16.73	174.82 ± 16.73	885.72 ± 11.13	**3.62 ± 3.35**	887.5 ± 10.28	**0.72 ± 0.8**	887.32 ± 9.60	**0.48 ± 0.64**
P-Value	$\mathbf{1.82 * 10^{-35}}$	$\mathbf{8.65 * 10^{-78}}$	**0.04**	0.32	**0.01**	**0.5**	0.02	0.003

1. Experiment 3.1 aims to compare the AVDHLA to the FSLA to observe the difference between these automata in getting penalties and rewards.
2. Experiment 3.2 aims to compare the AVDHLA with the VSLA to identify their strengths and weaknesses.
3. Experiment 3.3 aims to compare the AVDHLA to the HLA, the state-of-the-art method in the hybrid structure realm.

For the above experiments, three tuples of (ζ, ψ) are considered: $(0.0002, 0.00002)$, $(0.0005, 0.0005)$, and $(0.00002, 0.0002)$ for scenarios 1 to 3 respectively.

Experiment 3.1. The proposed model should be compared to the FSLA in the State-dependent environment. The motivation is to analyze the effect of the ζ and ψ parameters introduced in Section 6.1.2.

To conduct this experiment, the inner VASLAs are L_{R-I} with $\lambda_1 = 0.01$ and $\lambda_2 = 0$. The initial depths considered are 1, 3, 5, and 7 for configurations 1 to 4, respectively.

The initial reward probability vector of the desired environment is $[0.9, 0.1]$. This means that, in the first stage, action 1 has a reward probability of 0.9, and action 2 has a probability of 0.1 to be rewarded. In the next stages, this reward probability vector increases or decreases through the values of ζ and ψ.

In this experiment, action 1 is favored during the first stage. After 1000 iterations, this action remains favorable, although ζ and ψ influence the reward probability vector. The results in Table 8 demonstrate the superiority of the AVDHLA over the FSLA in terms of TNR and TNAS across most configurations, except for Configuration 3. In Configuration 3, the initial depth is set at 5, which appears to be an appropriate value for the automaton. However, the AVDHLA explores more through action switching to identify the most optimal depth.

Experiment 3.2. This experiment aims to evaluate the proposed automaton in terms of TNR and TNAS. To achieve this goal, we consider $K = 5$ and $N = 4$ for the AVDHLA. For the VSLA, the experiment is set up exactly as in Eq. (16). The inner VASLAs have the same configurations as the VSLA (except for the number of actions), as shown in Table 9.

According to the results of this experiment, as shown in Table 10, the AVDHLA outperforms the VSLA in most configurations with respect to TNR and TNAS. When both λ_1 and λ_2 are set to 0, the AVDHLA essentially functions as the FSLA, making decisions less frequently, which results in more rewards and fewer action switches. Among the various configurations, L_{R-P} demonstrates superior performance due to its sensitivity to penalties. Consequently, receiving penalties in the environment effectively impacts the probability vector, enabling the AVDHLA to adjust the depth appropriately. Overall, the AVDHLA proves to be mostly superior to the VSLA in state-dependent environments based on the defined experimental metrics.

Experiment 3.3. This experiment is designed to compare the AVDHLA with the HLA. The conducted environment is entirely similar to what has been designed for Experiment 3.1. The results are shown in Table 11. According to the results, the AVDHLA outperforms the HLA in terms of TNR and TNAS. It is evident that whenever the depth of the FSLA is not appropriate, due to the changing attributes of the environment, the AVDHLA will choose a depth in a way that maximizes its reward.

7. Application : Defense against selfish mining attack

In this section, the AVDHLA is used to design a novel defense mechanism, named Nik [53,84,85], against the selfish mining [52,86,87] attack in Bitcoin [51,88,89]. First, the related concepts such as the mining process and the selfish mining attack are briefly introduced. Then, the experiments performed to evaluate the performance of the proposed learning automaton in such a complex environment are presented.

Table 9
The setup of the VSLA in Experiment 3.2.

	P	L_{R-I}		L_{P-I}		L_{R-P}		$L_{R-\epsilon P}$	
	Config1	Config2.a	Config2.b	Config3.a	Config3.b	Config4.a	Config4.b	Config5.a	Config5.b
K	5	5	5	5	5	5	5	5	5
N	4	4	4	4	4	4	4	4	4
λ_1	0	0.1	0.01	0	0	0.1	0.01	0.1	0.01
λ_2	0	0	0	0.1	0.01	0.1	0.01	0.01	0.001

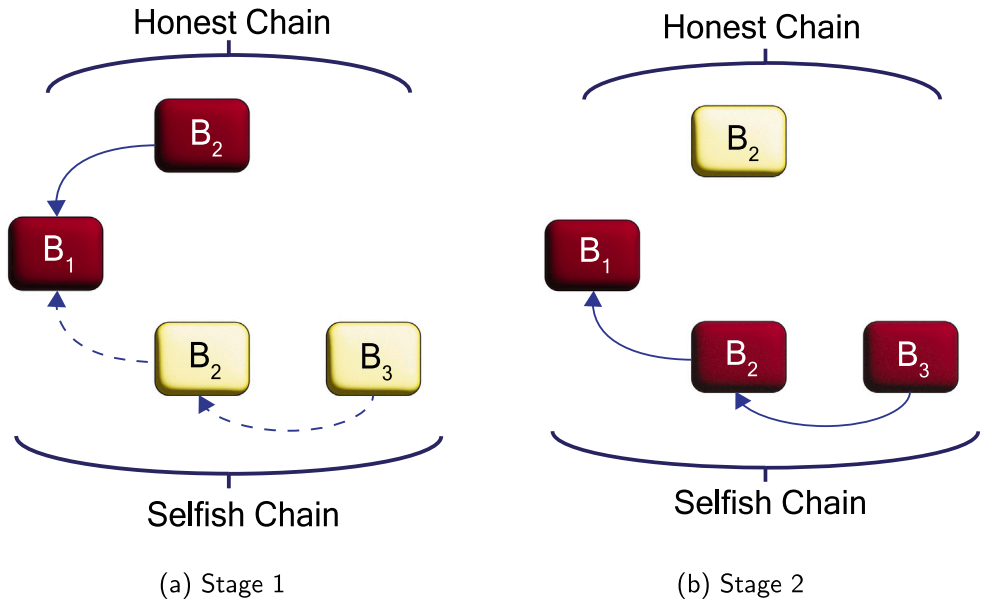


Fig. 11. A scenario of the selfish mining attack.

7.1. Selfish mining concepts

Bitcoin [51] is a decentralized cryptocurrency that was introduced by Satoshi Nakamoto in 2009. It has received a lot of attention due to its decentralized nature [51,88].

The transactions in the Bitcoin network are recorded through units called blocks. Creating a new block requires effort to solve a crypto puzzle with a dedicated reward. The participants who put their resources into solving such a puzzle are called miners [51–53,88].

The incentive mining process guarantees the safety of Bitcoin. It means each miner will be rewarded based on the shared resource. If this assumption works correctly, the decentralization of the network will be sustained [88].

However, handling Bitcoin in a decentralized manner is a challenging problem. Some attacks like selfish mining [52] threaten the most important property of Bitcoin by keeping newly discovered blocks private and revealing them whenever they get rewards more than their fair share.

Revealing kept blocks by selfish miners will make a fork in the chain of blocks. Under such circumstances, the honest branch of the fork, which is the result of the right work, will be discarded. Consequently, a consensus is reached on the selfish branch of the fork [90,91].

To enhance our comprehension of selfish mining, let us consider a simplified scenario. In the first stage, selfish miners aim to conduct their mining operations covertly, holding two blocks (B_2, B_3) ahead of the honest miners in secret, depicted by the yellow blocks in Fig. 11(a) within the selfish chain. When the honest miners discover the block B_2, shown in red in Fig. 11(a), they assume it is accepted and part of the main chain. However, this assumption is incorrect. The selfish miners then reveal their private chain, resulting in the work done by the honest miners being discarded, as illustrated by the yellow block in the honest chain in Fig. 11(b). In this scenario, the network adopts the secret blocks mined by the selfish miners, transforming the previous white blocks into red blocks in Fig. 11(b).

Our novel approach with the proposed learning automaton will overcome this frustrating problem in Bitcoin. To reach the ultimate goal, we can reduce this problem to a simpler problem of making a decision among the created branches of the fork in each distributed miner.

7.2. Proposed AVDHLA-based defense

In this section, we will describe our novel defense mechanism which takes advantage of a powerful learning automaton. A learning automaton serves as a decision-maker to increase safety in each node. It requires predefined criteria to assist the node in accurately selecting one branch of the fork, even if some branches are produced by selfish miners. These criteria are based on the characteristics of each branch in the fork, as follows:

- The length of a branch, denoted by L, is the number of blocks in that branch.
- The weight of a branch, denoted by W, is calculated as follows: Starting from the first block until the last one of that branch, it is compared with the blocks with the same height of the other branches. The weight of the branch with the most recent creation time will be increased by one in each iteration.
- The fail-safe parameter, denoted by ρ, will help the miner to choose a branch based on L or W. If the length of one branch in that fork is longer than the others by ρ, that branch will be chosen. Otherwise, the W parameter will be considered for choosing the fork.
- The decision-making time, denoted by τ, is the time for a miner to check if any forks exist. If such a fork exists, the miner should choose one of them, considering the ρ parameter.

Table 10
Experimental results of Experiment 3.2 with respect to TNR and TNAS.

Model	Config1		Config2.a		Config2.b		Config3.a		Config3.b		Config4.a		Config4.b		Config5.a		Config5.b	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
Scenario 1 - ($\zeta = 0.0002, \psi = 0.00002$)																		
AVDHLA	**655.42 ± 31.16**	**87.62 ± 53.11**	658.04 ± 38.10	87.34 ± 60.58	**657.08 ± 34.02**	**84.74 ± 57.40**	**655.08 ± 35.78**	**92.34 ± 59.37**	**661.46 ± 31.55**	**71.76 ± 44.87**	**660.92 ± 36.55**	**82.18 ± 64.55**	**650.34 ± 34.26**	**100.36 ± 61.02**	653.78 ± 38.16	88.16 ± 64.88	**656.24 ± 32.58**	**87.22 ± 52.94**
VSLA	187.22 ± 13.14	800.08 ± 11.98	**687.72 ± 14.31**	**28.74 ± 9.08**	580.34 ± 19.30	295.14 ± 33.03	213.9 ± 13.02	811.66 ± 10.23	208.18 ± 12.65	797.86 ± 11.58	396.3 ± 27.68	667.02 ± 27.93	357.54 ± 26.20	717.96 ± 22.36	**666.0 ± 14.73**	117.24 ± 19.80	555.8 ± 22.54	358.18 ± 32.21
P-Value	$\mathbf{4 * 10^{-99}}$	$\mathbf{9.34 * 10^{-97}}$	$\mathbf{1.63 * 10^{-6}}$	$\mathbf{1.35 * 10^{-9}}$	$\mathbf{1.45 * 10^{-24}}$	$\mathbf{4.49 * 10^{-40}}$	$\mathbf{1.21 * 10^{-91}}$	$\mathbf{6.64 * 10^{-93}}$	$\mathbf{1.48 * 10^{-97}}$	$\mathbf{2.37 * 10^{-104}}$	$\mathbf{6.70 * 10^{-63}}$	$\mathbf{8.17 * 10^{-78}}$	$\mathbf{1.78 * 10^{-69}}$	$\mathbf{2.34 * 10^{-83}}$	**0.03**	**0.003**	$\mathbf{2.30 * 10^{-32}}$	$\mathbf{5.95 * 10^{-52}}$
Scenario 2 - ($\zeta = 0.0005, \psi = 0.0005$)																		
AVDHLA	**568.7 ± 16.18**	**120.14 ± 37.49**	**564.36 ± 19.45**	129.14 ± 51.15	567.38 ± 15.73	**122.1 ± 37.27**	**567.2 ± 18.10**	**130.48 ± 32.85**	**575.14 ± 18.48**	**113.26 ± 34.90**	**564.96 ± 18.45**	**127.78 ± 47.45**	**565.76 ± 16.61**	**130.88 ± 39.33**	567.84 ± 17.47	**122.36 ± 47.05**	564.66 ± 16.13	**124.94 ± 29.11**
VSLA	346.72 ± 13.43	798.1 ± 11.99	552.52 ± 15.20	**32.4 ± 11.37**	**577.66 ± 12.38**	326.48 ± 38.12	380.0 ± 12.88	806.64 ± 12.95	374.36 ± 15.93	798.02 ± 11.26	510.4 ± 15.50	645.28 ± 22.40	508.4 ± 18.55	695.54 ± 22.51	**579.64 ± 14.24**	266.4 ± 34.0	**570.86 ± 12.60**	407.82 ± 32.35
P-Value	$\mathbf{9.71 * 10^{-88}}$	$\mathbf{2.39 * 10^{-108}}$	**0.001**	$\mathbf{6.88 * 10^{-23}}$	**0.0005**	$\mathbf{6.03 * 10^{-47}}$	$\mathbf{2.30 * 10^{-78}}$	$\mathbf{7.97 * 10^{-113}}$	$\mathbf{2.20 * 10^{-77}}$	$\mathbf{9.25 * 10^{-112}}$	$\mathbf{8.77 * 10^{-29}}$	$\mathbf{6.67 * 10^{-85}}$	$\mathbf{2.58 * 10^{-29}}$	$\mathbf{1.07 * 10^{-94}}$	**0.0004**	$\mathbf{1.13 * 10^{-31}}$	**0.02**	$\mathbf{1.04 * 10^{-67}}$
Scenario 3 - ($\zeta = 0.00002, \psi = 0.0002$)																		
AVDHLA	**771.08 ± 28.87**	**29.3 ± 40.52**	**780.58 ± 18.58**	**16.72 ± 17.16**	**776.98 ± 19.13**	**25.1 ± 26.34**	**779.98 ± 19.60**	**18.62 ± 20.10**	**773.62 ± 23.96**	**25.92 ± 31.86**	**769.68 ± 31.06**	**29.54 ± 36.01**	**769.44 ± 28.48**	**32.8 ± 33.49**	**772.92 ± 34.23**	**29.62 ± 42.0**	**773.12 ± 24.73**	**26.06 ± 27.47**
VSLA	279.02 ± 15.93	799.94 ± 10.82	773.96 ± 11.78	29.4 ± 8.76	664.78 ± 18.14	306.28 ± 28.22	307.5 ± 12.45	810.7 ± 13.72	305.36 ± 13.93	798.9 ± 11.39	534.2 ± 28.50	579.74 ± 34.19	480.7 ± 29.26	679.72 ± 32.69	757.32 ± 16.23	96.42 ± 19.27	651 ± 19.86	347.22 ± 28.88
P-Value	$\mathbf{2.75 * 10^{-102}}$	$\mathbf{4.38 * 10^{-111}}$	**0.03**	$\mathbf{1.23 * 10^{-5}}$	$\mathbf{6.65 * 10^{-51}}$	$\mathbf{2.31 * 10^{-72}}$	$\mathbf{2.46 * 10^{-135}}$	$\mathbf{2.18 * 10^{-115}}$	$\mathbf{1.54 * 10^{-107}}$	$\mathbf{2.60 * 10^{-120}}$	$\mathbf{1.36 * 10^{-61}}$	$\mathbf{9.02 * 10^{-90}}$	$\mathbf{3.87 * 10^{-71}}$	$\mathbf{4.57 * 10^{-99}}$	**0.004**	$\mathbf{6.73 * 10^{-17}}$	$\mathbf{6.28 * 10^{-47}}$	$\mathbf{1.63 * 10^{-76}}$

Table 11
Experimental results of Experiment 3.1 with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
Scenario 1 - ($\zeta = 0.0002, \psi = 0.00002$)								
AVDHLA	**788.16 ± 12.00**	**32.88 ± 17.55**	**794.58 ± 15.37**	**21.5 ± 15.54**	**802.0 ± 11.55**	**3.0 ± 4.27**	**802.8 ± 10.94**	**0.94 ± 1.51**
HLA	752.66 ± 15.31	248.02 ± 15.28	786.1 ± 13.25	32.96 ± 12.18	798.6 ± 11.74	5.9 ± 4.07	796.36 ± 10.98	2.46 ± 1.59
P-Value	$\mathbf{1.41 * 10^{-22}}$	$\mathbf{3.40 * 10^{-82}}$	**0.004**	$\mathbf{9.82 * 10^{-5}}$	0.15	**0.0008**	**0.004**	$\mathbf{4.76 * 10^{-6}}$
Scenario 2 - ($\zeta = 0.0005, \psi = 0.0005$)								
AVDHLA	**657.52 ± 12.32**	**84.86 ± 24.22**	**662.5 ± 13.30**	**70.16 ± 27.25**	**664.1 ± 11.91**	**42.18 ± 17.24**	**661.3 ± 13.92**	**27.16 ± 11.68**
HLA	656.02 ± 13.52	344.54 ± 13.58	656.96 ± 13.78	120.7 ± 17.97	656.5 ± 13.33	70.12 ± 20.06	657.04 ± 15.33	46.42 ± 18.37
P-Value	0.56	$\mathbf{1.08 * 10^{-82}}$	**0.04**	$\mathbf{1.87 * 10^{-18}}$	**0.003**	$\mathbf{4.87 * 10^{-11}}$	0.15	$\mathbf{1.39 * 10^{-8}}$
Scenario 3 - ($\zeta = 0.00002, \psi = 0.0002$)								
AVDHLA	**884.44 ± 12.63**	**17.98 ± 11.92**	**889.36 ± 9.84**	**3.94 ± 4.70**	**891.16 ± 8.80**	**0.86 ± 0.95**	**892.64 ± 7.75**	**0.58 ± 0.75**
HLA	847.8 ± 15.02	153.02 ± 15.06	884.7 ± 9.98	6.24 ± 4.14	887.66 ± 9.17	2.54 ± 2.23	888.96 ± 9.88	2.8 ± 2.42
P-Value	$\mathbf{3.50 * 10^{-23}}$	$\mathbf{6.79 * 10^{-71}}$	**0.022**	**0.011**	**0.05**	$\mathbf{5.04 * 10^{-6}}$	**0.04**	$\mathbf{1.91 * 10^{-8}}$

Algorithm 3 UpdateFailSafe(LA, ρ, ρ_{min}, ρ_{max})

```
Begin
  if (ρ = ρ_max) then
    L = LA.choose _ action(['Stop', 'Shrink'])
  else if (ρ = ρ_min) then
    L = LA.choose _ action(['Grow', 'Stop'])
  else
    L = LA.choose _ action(['Grow', 'Stop', 'Shrink'])
  end if
  switch (L)
  case 'Grow':
    ρ = ρ + 1
  case 'Shrink':
    ρ = ρ − 1
  case 'Stop':
    /*Do nothing about ρ*/
  end switch
End
```

- The time parameter, denoted by θ, is used to configure the next value of ρ using the automaton. Each θ consists of several τ.

The considered algorithm to aid the miner in decision-making when faced with multiple branches operates as follows, and its corresponding flowchart is presented in Fig. 12:

1. Calculate L for each branch.
2. Calculate W for each branch.
3. After sorting by the longest chain, if the difference between the length of the longest and second longest branch is greater than ρ, the miner must choose the longest branch. Otherwise, it must choose the heaviest chain according to the calculated W parameter.

The following algorithm relates to scenarios requiring the modification of the fail-safe parameter and details how the learning automaton receives feedback. Its corresponding flowchart is illustrated in Fig. 13:

1. If τ reaches its end, the learning automaton should choose the next value for ρ. Usually, ρ swings between ρ_{min} and ρ_{max}. The learning automaton has three options: 1- **Grow** to increase ρ by one, 2- **Stop** to leave ρ unchanged, 3- **Shrink** to decrease ρ by one. This algorithm updates the fail-safe parameter, which is shown in Algorithm 3.
2. If θ reaches its end, the learning automaton should receive feedback from the environment. We have designed a virtual environment for the learning automaton to provide information about its decision. The reinforcement signal is calculated by dividing the number of decisions made based on W by the total number of decisions. The total number of decisions consists of the number of decisions based on height (L) plus the number of decisions based on weight (W). The following equation demonstrates the β parameter of the learning automaton.

$$\beta = \frac{Number\ of\ Weight\ Decisions}{Total\ Number\ of\ Decisions} \tag{20}$$

7.3. Selfish mining metrics

Two metrics are considered to evaluate the performance of the learning automaton against the selfish mining attack, as well as what is defined in Section 6.2. These metrics are:

1. **Relative Revenue**: This metric is used to measure the revenue of a miner based on the revenue of others. In fact, we want to assess the margin between the revenue of the miner when the selfish attack is performed in the network and when the network has the incentive-compatibility property. This metric is calculated as follows:

$$\frac{Number\ of\ Mined\ Block\ by\ i^{th}\ Miner}{Total\ Number\ of\ Mined\ Blocks} \tag{21}$$

2. **Lower Bound Threshold**: This metric evaluates the minimum computing power that a selfish miner should provide to start an attack.

7.4. Proposed defense experiment

This experiment aims to provide a comparison of the proposed defense using a more sophisticated learning automaton with the well-known defense, namely tie-breaking. In tie-breaking, when a miner is aware of the fork, they choose one of the branches of that fork in a uniform random manner.

We assume that there are two categories of miners in the network. All the miners who obey Bitcoin protocols form an honest pool. By contrast, all the selfish miners together form a selfish pool. We prefer to study the performance of the defense from the selfish pool's point of view.

The proposed algorithm was simulated by transforming the mining model into a Monte Carlo simulation process. This adaptation enables the equitable distribution of newly discovered blocks among selfish and honest miners without the need to solve a cryptographic puzzle. For reproducibility, the simulator developed for this purpose is available on GitHub.[2] The experiments detailed in this subsection were executed on an Intel Core i7 processor with a clock frequency of 2.5 GHz.

[2] https://github.com/AliNikhalat/SelfishMining.

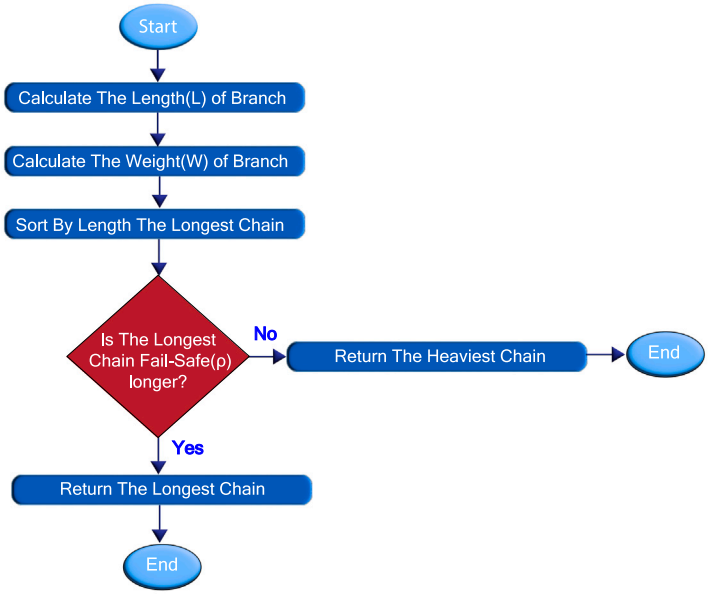


Fig. 12. The flowchart of the algorithm to aid the miner in decision-making.

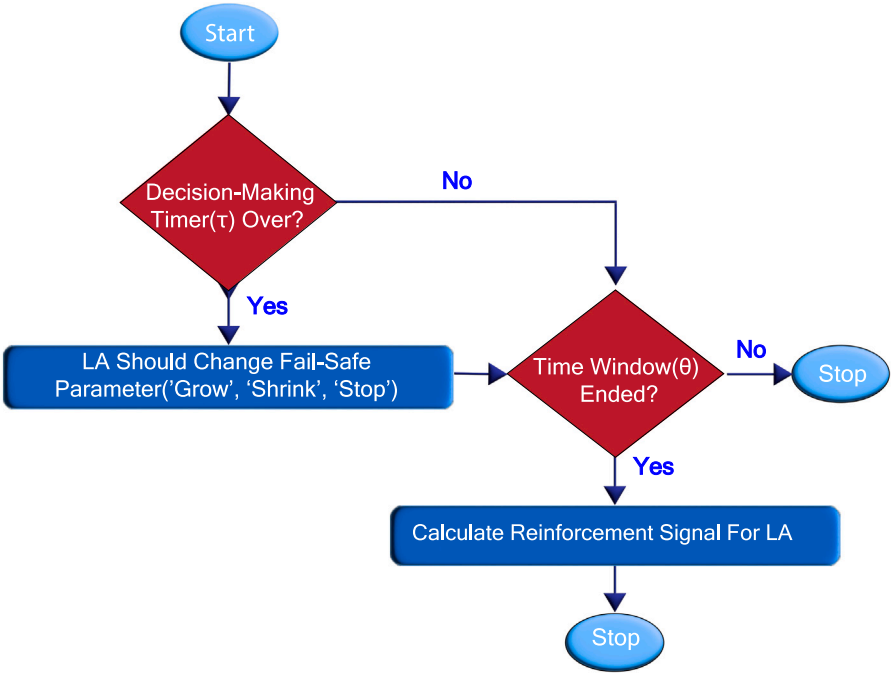


Fig. 13. The flowchart of modifying fail-safe parameter and how the learning automaton receives feedback.

To accomplish our objective, we generated 10000 blocks to ensure that the learning agents have a sufficient number of iterations to comprehend the events in the network. The parameter ρ can vary between $\rho_{min} = 1$ and $\rho_{max} = 5$. This range for ρ was selected to enable the learning automaton to reach consensus quickly, considering that choosing ρ is a time-consuming process. A large gap between ρ_{min} and ρ_{max} could potentially lead to inconsistencies between each node's learning automaton in the network. Also, five configurations for the VASLA are considered: 1- P model with $\lambda_1 = 0$ and $\lambda_2 = 0$, 2- L_{R-I} model with $\lambda_1 = 0.01$ and $\lambda_2 = 0$, 3- L_{P-I} model with $\lambda_1 = 0$ and $\lambda_2 = 0.01$, 4- L_{R-P} with $\lambda_1 = 0.01$ and $\lambda_2 = 0.01$, 5- $L_{R-\epsilon P}$ with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$.

The results in Fig. 14 show interesting points. Firstly, they show the superiority of the proposed defense over tie-breaking with respect to the relative revenue of the selfish miners. Secondly, among the various kinds of automata, the AVDHLA outperforms the others in conjunction with relative revenue. Both pieces of evidence lead us to the outstanding performance of the automaton in a complex environment like the blockchain. Additionally, it shows that the AVDHLA adopts each depth of predefined actions correctly to get a higher reward.

Besides the relative revenue, the other metric, namely the lower bound threshold, is studied. The results, which are demonstrated in Table 12, reveal interesting points once again from the other side. The proposed defense using the power of the AVDHLA increases the lower bound threshold from 0.25 up to 0.4 approximately. The AVDHLA tries to get rewards based on the forks created in the network. It can detect when a fork needs a decision by the weight of the height parameter precisely. As a matter of fact, this decision in the AVDHLA is made

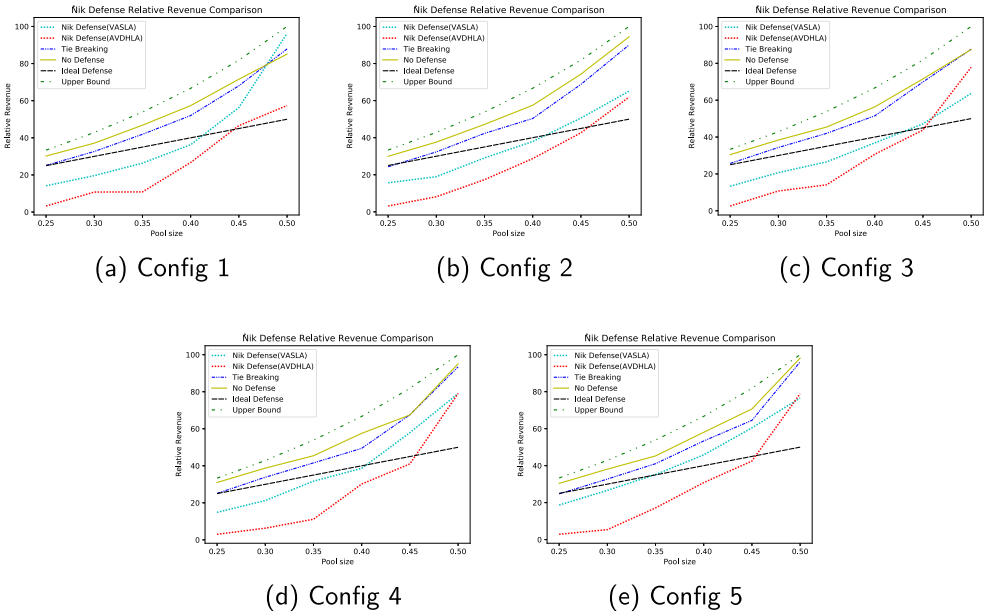


Fig. 14. The performance of Nik defense in compare to tie-breaking with respect to relative revenue.

Table 12
Experimental results of the Nik defense with respect to the lower bound threshold.

Defense	P	L_{R-I}	L_{P-I}	L_{R-P}	$L_{R-\varepsilon P}$
Nik (AVDHLA)	**0.45**	**0.46**	**0.46**	**0.46**	**0.46**
Nik (FSLA)	**0.42**	**0.41**	**0.44**	**0.41**	**0.36**
Tie-breaking	0.25	0.25	0.25	0.25	0.25

from the actual understanding of the automaton from an unknown environment like the blockchain. Eventually, we can trust the AVDHLA as a decision-maker in the blockchain networks.

Eventually, we can trust the AVDHLA as a decision-maker in the blockchain networks. Also, some extra experiments are designed to investigate the effect of other parameters on the quality of defense concerning the introduced metrics in Appendix D.

8. Application : Recommendation systems

In this section, the AVDHLA is utilized to develop a novel adaptive recommendation method for personalized content delivery. For the first time in the literature, we apply the fixed structure learning automata family to a recommender system [92,93], evaluating its performance in a real-world application. We begin by introducing the fundamental concepts behind recommendation systems and their evaluation metrics, followed by a detailed explanation of the adaptive recommendation algorithm built on learning automata.

In the realm of personalized content delivery, recommender systems play a critical role by providing tailored suggestions based on user preferences. These systems have become indispensable in various domains [94,95] such as streaming services [96], e-commerce [97], and social media [98]. The primary objective of a recommender system is to predict and serve the most relevant content to each user by analyzing their historical behavior and interactions. Broadly speaking, recommender systems are categorized into three main types [99]: content-based, collaborative filtering, and hybrid methods. Content-based systems recommend items that share features with previously liked content, while collaborative filtering leverages the behavior and preferences of similar users to make predictions. Hybrid approaches combine both techniques to achieve a more accurate and adaptable recommendation, often addressing the limitations of individual methods.

One of the most significant challenges faced by recommender systems is managing the balance between exploration and exploitation. Exploration involves recommending new or less-explored content to discover new user preferences, while exploitation focuses on delivering content that aligns with known user preferences based on past interactions. Striking this balance is crucial in dynamic environments where user interests evolve over time. As a result, adaptive algorithms that can learn in real-time and adjust recommendations accordingly have become a major focus in advancing recommender systems, ensuring that they remain effective in continuously changing contexts.

In our work, we leveraged the MAB2Rec[3] [100–102] library to build upon this foundation and enhance the adaptability of recommendation systems. MAB2Rec is a specialized library designed to implement multi-armed bandit (MAB) algorithms within the framework of recommender systems. MAB algorithms are particularly well-suited to addressing the exploration–exploitation dilemma, where each "arm" represents a different recommendation option. The system selects an arm at each iteration based on expected rewards, such as click-through rates or purchase likelihood, with the goal of maximizing cumulative rewards over time.

As part of our contribution, we applied the AVDHLA to the MAB2Rec framework [100–102], introducing an innovative adaptive mechanism for dynamically adjusting the exploration–exploitation trade-off. By doing so, we extended the capabilities of the MAB-based recommendation system, allowing it to continuously adjust its recommendation strategy in response to changing user behaviors. This integration of the AVDHLA into MAB2Rec led to improvements in key performance metrics, such as the Click-Through Rate (CTR) and Precision, demonstrating the practical versatility of our approach and its potential to significantly enhance the performance of real-world recommender systems.

CTR [103,104] is a key metric in evaluating the effectiveness of a recommendation system. It measures the proportion of times a user clicked on a recommended item, calculated as the number of clicks divided by the total number of recommendations presented to the user. This metric directly reflects how relevant the recommendations are to users, with a higher CTR indicating more successful engagement.

[3] https://fidelity.github.io/mab2rec/.

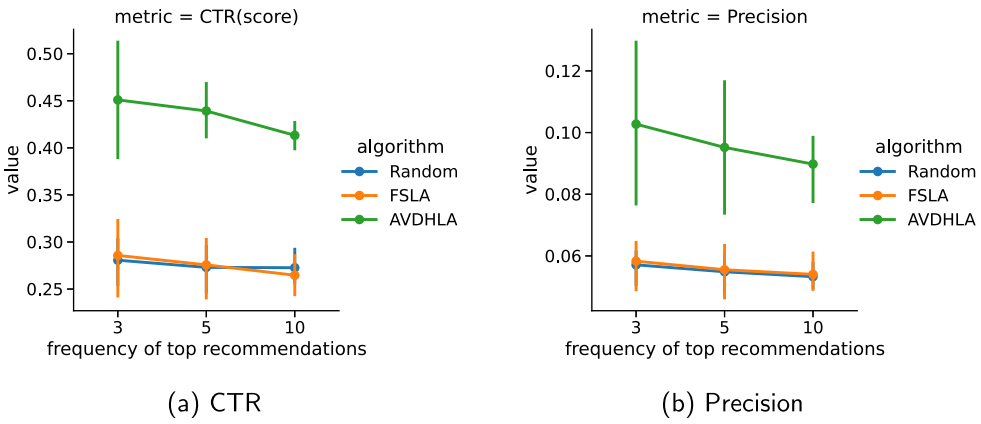


Fig. 15. Experimental results of recommendation systems for $N = 1$.

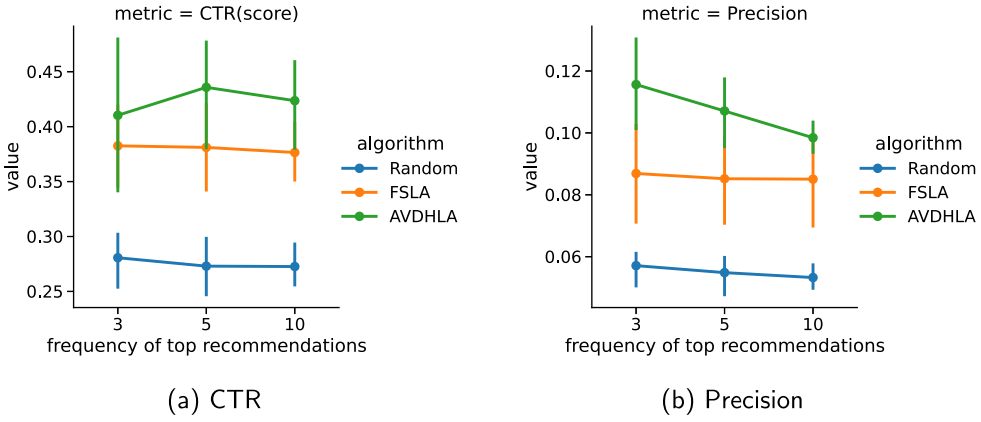


Fig. 16. Experimental results of recommendation systems for $N = 3$.

Precision [103,104] measures the ratio of relevant items recommended to the total number of items recommended. In recommendation systems, it reflects how many recommended items align with user preferences. Notably, a recommender system that serves recommendations to only 5% of users can still achieve high precision if those users consistently interact with the recommendations. High precision indicates that the system provides accurate and relevant suggestions, which is essential for improving user satisfaction and system performance.

To evaluate the proposed method, we utilized the MovieLens dataset,[4] which captures users' expressed preferences for movies. This dataset is split into training and test sets. During the training phase, the recommender system requires data on a set of users, items, and their interactions. For the MovieLens dataset, an interaction is marked as 1 if a user rates a movie as 5 on a 1–5 scale, and 0 otherwise. In the testing phase, the performance of the learned agent is assessed using CTR and precision metrics to measure its effectiveness.

Two depth levels, $N = 1$ and $N = 3$, were considered for both the AVDHLA and the FSLA in the evaluation. The internal VASLAs in the AVDHLA use the L_{R-I} configuration, with $\lambda_1 = 0.1$ and $\lambda_2 = 0$. The recommender system was tested by recommending the top 3, 5, and 10 movies to users, with the results compared against a completely random recommender using CTR and Precision metrics.

The results, illustrated in Figs. 15 and 16, demonstrate that the AVDHLA consistently outperforms the FSLA in terms of both CTR and Precision. The superior performance of the AVDHLA can be attributed to its ability to dynamically adjust its depth, optimizing the balance between exploration and exploitation. For $N = 1$, the FSLA tends to focus more on exploration due to the shallow depth, which limits its ability to exploit known information. However, by increasing the depth to $N = 3$, the FSLA improves performance through increased exploitation. Nevertheless, the AVDHLA shows a clear advantage as it avoids the limitations faced by the FSLA by adjusting depth more intelligently to adapt to different scenarios.

It is worth noting that precision rates of 5% are generally considered high for most recommender systems. In this case, the learning automaton-based recommender, particularly the AVDHLA, achieved precision rates around 10%, demonstrating excellent practical performance. As expected, both CTR and Precision decrease as the number of recommended items increases. This decline occurs because users typically concentrate on only a few top recommendations, and are less likely to engage with items beyond the top selections. Hence, recommending more items may result in a marginal increase in clicks, but overall engagement will likely decline.

9. Discussion

In this section, we go deeper into a comprehensive discussion of our paper, analyzing its strengths and weaknesses to show the applicability of our model across diverse scenarios. Additionally, we explore potential avenues for improvement. The ensuing points encapsulate our analysis:

- A key distinguishing feature of the proposed method, compared to its predecessors, lies in its ability to dynamically adjust its depth based on interactions with the surrounding environment. This adaptive capability enables our approach to excel in balancing effective exploration and exploitation.

[4] https://grouplens.org/datasets/movielens/100k/.

- The implementation of the AVDHLA is straightforward and can be reproduced easily by combining $L_{KN,K}$ with a few VASLAs, as illustrated in Section 5.
- The AVDHLA is designed for intricate scenarios within complex environments, owing to its self-adaptive capabilities.
- The time complexity of our algorithm is anticipated to surpass that of the $L_{KN,K}$ due to the utilization of the VASLA at each depth. Consequently, the learning time is expected to increase. However, this investment in time is justified by the algorithm's effective tuning, ultimately leading to enhanced long-term rewards.

- The space complexity of our algorithm is expected to see a moderate increase compared to the $L_{KN,K}$, as each VASLA requires a 1-D vector with three inner items to store 'Grow', 'Shrink', and 'Stop' probabilities.
- Improperly configuring the reward and penalty rate parameters can significantly impact the performance of the AVDHLA, potentially leading to incorrect selection of the appropriate depth.

- While many implementations leverage reinforcement learning to maximize rewards beyond their equitable share, our approach contradicts this trend. Instead, we employ it to mitigate the effectiveness of potential attacks. Additionally, we uphold ethical considerations by abstaining from using any private data in the blockchain.
- It is worth noting that in certain environments, altering the action may introduce a bias in the AVDHLA towards a specific action, leading to a significant increase in the depth of that action. To address this challenge, a potential solution is to introduce a new parameter for controlling the maximum value of the depth.
- It is highly recommended that the AVDHLA be applied to secure data transmission and trustworthiness frameworks, such as the approach proposed by Jiang et al. [105] to counter cyber-physical attacks. The AVDHLA can function as a decision-maker, efficiently managing secure transmission and attack detection simultaneously.
- The proposed system is rooted in reinforcement learning, and its capabilities can be further expanded by incorporating neural networks and leveraging deep learning concepts, as explored in studies such as [106–109].
- Since prevailing solutions against selfish mining predominantly rely on reinforcement learning [90,91], it is worth considering alternative AI methods, such as those proposed in [110,111], to address this issue in line with contemporary trends.

10. Conclusion and future research directions

The greatest weakness of the $L_{KN,K}$ learning automaton, as the most prominent member of the FSLA family, is the problem of choosing an appropriate depth. This problem has remained unsolved since the appearance of the $L_{KN,K}$. The lack of an effective solution has caused this type of learning automaton to fade in recent years. Therefore, we were motivated to solve this problem using an intelligent self-adaptive model named AVDHLA.

We have uncovered the critical importance of the initial depth setting in the FSLA interaction with its environment. By allowing the learning automaton's depth to adapt dynamically during action switching, even when the initial depth is not appropriately configured, the FSLA demonstrates the ability to autonomously adjust to unknown environments. To achieve this adaptability, we conducted another reinforcement learning agent, VASLA, into the decision-making process. Our findings indicate that over the long term, a well-set depth remains unchanged, while an incorrectly set depth undergoes modifications, improving the learning automaton's chances of attaining maximum rewards.

The proposed model was evaluated in various aspects. From the perspective of getting more rewards and fewer penalties, it outperformed the FSLA in different environments, including stationary, non-stationary environments such as Markovian switching, and State-dependent environments.

In addition to the predefined environments, we tried to apply the new model in an uncertain environment like blockchain. We designed a new strategy against the selfish mining attack, which threatened the incentive-compatibility of Bitcoin. In that strategy, the AVDHLA performed successfully compared to the FSLA in the designed experiments.

Consequently, the proposed model can be a robust alternative to the FSLA. Since the FSLAs have been used in a wide range of applications in different areas, the AVDHLA can also be used instead of them with just a bit of modification.

We are hopeful that the AVDHLA will promise a bright future ahead for AI. However, several interesting questions warrant exploration in future research endeavors. Key areas include establishing the learning ability and convergence of the AVDHLA to optimal actions, conducting a comprehensive analysis of its time and space complexity, and undertaking mathematical examinations of AVDHLA's performance across various environmental conditions.

Looking ahead, a centralized agent could potentially assume control over the depth of each action, offering a centralized approach to depth management. Additionally, there is potential for a more unified approach by combining the selection of the next action with the depth selection problem, thereby empowering a singular, intelligent agent to orchestrate both seamlessly.

CRediT authorship contribution statement

Ali Nikhalat-Jahromi: Writing – review & editing, Writing – original draft, Validation. **Ali Mohammad Saghiri:** Supervision. **Mohammad Reza Meybodi:** Supervision.

Funding

This research received no external funding.

Declaration of competing interest

We affirm that this research was conducted and financed solely using personal resources, with no external funding sources associated with this study.

We wish to confirm that there are no conflicts of interest that could potentially affect the objectivity, impartiality, or credibility of the research findings presented in this study.

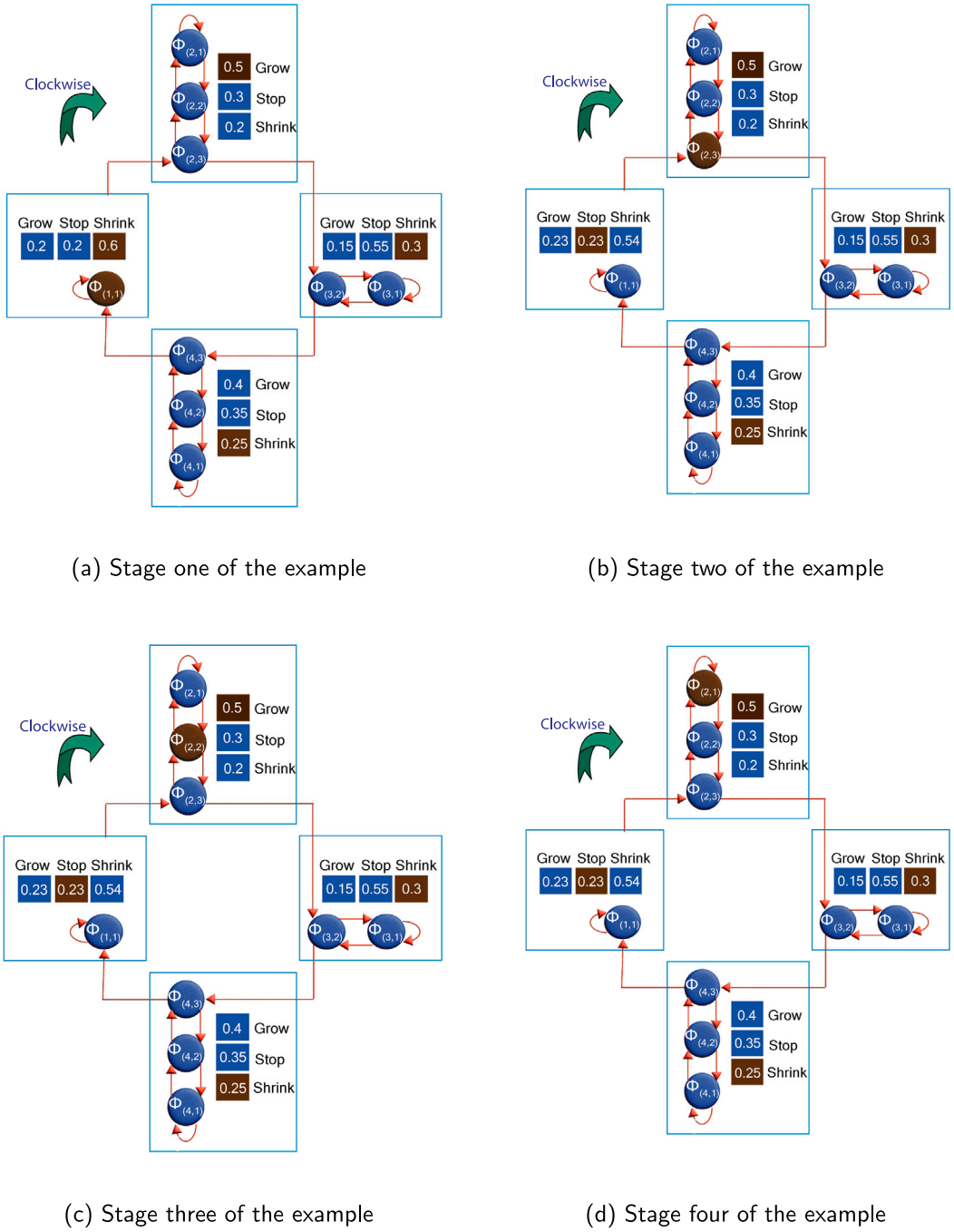
Throughout the research process, we have maintained complete independence and autonomy in the design, execution, and interpretation of the study's results.

Our primary objective in undertaking this research was to contribute to the advancement of knowledge in the field and address the research questions without any external influences or competing interests.

We assure transparency and credibility in reporting the research findings, strictly adhering to ethical principles and guidelines throughout the study.

Given the absence of any external funding, we are fully committed to ensuring the integrity and accuracy of the research. We pledge to promptly disclose any potential conflicts of interest that may arise during the course of this study.

This Declaration of Interest Statement serves as our genuine confirmation of the absence of external funding and conflicts of interest in the research project.



(a) Stage one of the example

(b) Stage two of the example

(c) Stage three of the example

(d) Stage four of the example

Fig. A.17. Four stages of an illustrative example for the AVDHLA.

Appendix A. An illustrative example

To provide an example for the proposed learning automaton, we consider a scenario in which after passing some iterations from the initial state, the AVDHLA came to the state demonstrated in Fig. A.17.

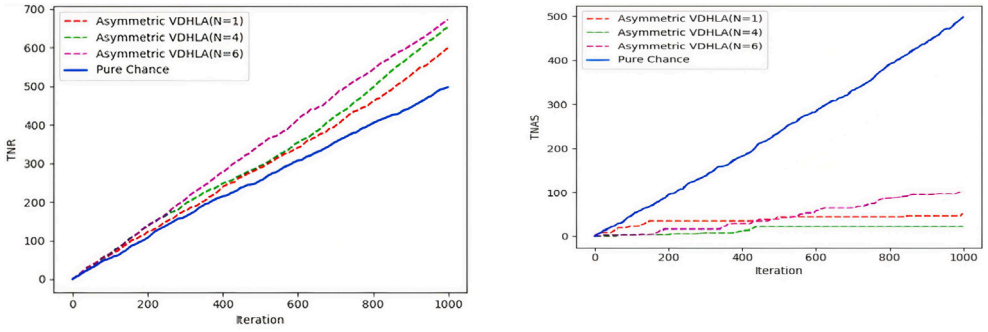
This scenario has four stages. The used AVDHLA is constructed with $AVDHLA(K = 4, N_1 = 1, N_2 = 3, N_3 = 2, N_4 = 3, \lambda_1 = 0.5, \lambda_2 = 0.5)$. This learning automaton has 4 actions. Each action has its own depth (1 for action 1, 3 for action 2, 2 for action 3, and 3 for action 4). Also, each action has a specific VASLA with a particular probability vector. The VASLAs are L_{R-P} with $\lambda_1 = \lambda_2 = 0.1$. In Fig. A.17, the last action made by each VASLA is marked.

At the first stage, the AVDHLA is in the state $\phi_{(1,1)}$, and the last chosen action by the conducted VASLA, denoted by VASLA_1, is 'Shrink'. We suppose that the learning automaton needs to be penalized. This state is shown in Fig. A.17.a.

In the second stage, since the FSLA is on the $\phi_{(1,1)}$, which is an edge state, the first step is to investigate the next action for the FSLA. Due to the clockwise policy of the FSLA, it will choose the state $\phi_{(2,3)}$.

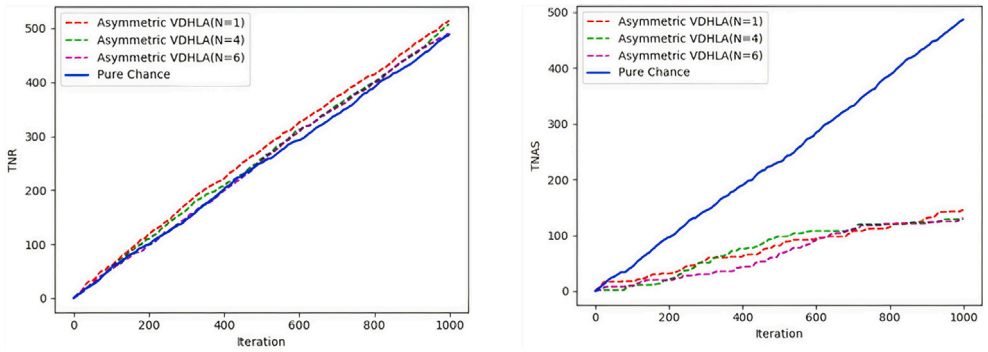
In this situation, the VASLA_1 should choose the next action too. This action-selection relates to the next depth of action 1. But before choosing the next action, the probability vector of VASLA_1 needs to be updated.

Since the depth of action 1 is 1, each transition in this action is the depth transition. Following the assumption, there were no transitions in this action. So β of the VASLA_1 is 0. This shows that the last action was not good.



(a) TNR of AVDHLA (b) TNAS of AVDHLA

Fig. A.18. Experimental results of Experiment 1.1 with reward probability vector of (0.3, 0.7).



(a) TNR of AVDHLA (b) TNAS of AVDHLA

Fig. A.19. Experimental results of Experiment 1.1 with reward probability vector of (0.5, 0.5).

The probability vector of the VASLA$_1$ will transit from $[0.2, 0.2, 0.6]$ to $[0.23, 0.23, 0.54]$. This transition is the result of updating with the Eq. (1) for the 'Shrink' action, and the Eq. (2) for the other actions. These changes are applied to Fig. A.17.b, which is related to the current stage.

The VASLA$_1$ needs to omit the 'Shrink' action from its options for choosing the next action because the depth of action 1 is equal to 1, and if the 'Shrink' action is chosen, the next depth will be 0, leading to an invalid result. Under such circumstances, the VASLA$_1$ undertakes to choose between the 'Grow' or 'Stop' options. We assume that the 'Stop' option is chosen by the VASLA$_1$.

In the third stage, the learning automaton performs action number 2 and gets a reward from the outer environment, after which it goes to the state $\phi_{(2,2)}$. The $transition - counter$ variable increases by one, but the action was not a depth-transition. So, the $depth - transition$ remains 0 (Fig. A.17.c).

In the fourth stage, the AVDHLA performs action number 2 again, and this time the action is successful. The FSLA changes the state from $\phi_{(2,2)}$ to $\phi_{(2,1)}$. Not only will the $transition - counter$ increase by one, but also $depth - transition$ will increase its value to reach 1. Fig. A.17.d displays the fourth stage.

To have a better overview, it is repeated once again. The FSLA deserves the reward from the environment, remains in state $\phi_{(2,1)}$. Similar to stage 4, both the $transition - counter$ and $depth - transition$ will increase because there is a transition, and that transition is of the depth-transition kind. Fig. A.17.d depicts the status of the learning automaton in the last stage too.

This scenario will continue, and as can be seen, the proposed learning automaton can synchronize itself with the surrounding environment to obtain the highest possible number of rewards.

Appendix B. Exploring the stationary environment: Additional experiment insights

B.1. Experiment 1.1 extras

Extra experiments were designed to learn more about the capabilities of the new learning automaton. This time, the probability of getting a reward of α_1 was increased up to 0.3, while α_2 was decreased to 0.7. For the third time, a totally random environment was considered with the probability of getting a reward for both actions being 0.5 equally. Results of these experiments are shown in Figs. A.18 and A.19 respectively.

The results of both figures show that both of them receive rewards more than the pure chance automaton. On the other hand, they have a lot fewer number of action switching. This leads us to the clear capability of learning in the AVDHLA.

As evident in Fig. A.18(a), when the environment favors an action with a probability of 0.7, a depth value of 6 yields more rewards compared to depths of 4 and 1. However, as showcased in the dynamic nature of depth changes in the AVDHLA illustrated in Fig. A.18(b), the learning automaton undergoes action switching and depth adjustments in its pursuit of finding an appropriate depth. Furthermore, the learning automaton with $N = 1$ experiences depth changes aimed at increasing its depth.

In Fig. A.19(a), given equal probabilities for both actions, the learning automaton with the lowest depth exhibits superior adaptability to the environment, resulting in higher rewards. Additionally, Fig. A.19(b)

Table B.13
Experimental results of Experiment 1.2.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
				K = 2				
AVDHLA	**8032**	**31**	**8075**	**25**	**8045**	**8**	**8000**	**1**
FSLA	6736	3264	7986	142	7953	31	8000	4
				K = 5				
AVDHLA	**7934**	**123**	**8018**	**37**	**7990**	**26**	**8008**	**7**
FSLA	4600	5400	7667	475	7980	45	8002	4

demonstrates that the lowest depth experiences more frequent action switching in its quest to identify an appropriate depth.

Besides, we should mention that the greater the difference between the probability of getting rewards between two actions (like the (0.9, 0.1) probability vector in Experiment 1.1), the better the AVDHLA can adapt itself to the surrounding environment. Because the AVDHLA increases its depth decisively, as a result, it does not change the chosen action easily.

B.2. Experiment 1.2 extras

Completing our comparison of the AVDHLA family with the FSLA requires more experiments. To accomplish this, Experiment 1.2 is repeated for learning automata with 2 and 5 actions. The results are shown in Table B.13.

These results show the superiority of the AVDHLA over the FSLA with respect to TNR and TNAS. Lower values of K can help the learning automaton find the favorable action more easily, therefore TNR is higher than in an experiment with a higher number of K. Also, this has an effect on reducing the total number of action switching because the learning automaton finds the favorable action among the lower number of actions more simply.

Appendix C. Exploring the Markovian switching environment: Additional experiment insights

Several extra experiments are designed to assess the performance of the proposed learning automaton in a Markovian switching environment to compare it with the FSLA with respect to TNR and TNAS metrics.

For a better understanding of the traits of the AVDHLA in a Markovian switching environment, four scenarios of Fig. C.20 with a simple two-state Markov chain are considered as follows:

1. This scenario is devoted to a Markovian switching environment with two states that tend to stay in the current state, and switching from one state to the other causes the favorable action to change. More details are explained in Appendix C.1.
2. This scenario is devoted to a Markovian switching environment with two states that tend to change the current state, and switching from one state to the other causes the favorable action to change. More details are explained in Appendix C.2.
3. This scenario is devoted to a Markovian switching environment with two states that tend to stay in the current state, and switching from one state to the other just decreases the probability of the favorable action, but the favorable action is the same as before. More details are explained in Appendix C.3.
4. This scenario is devoted to a Markovian switching environment with two states that tend to change the current state, and switching from one state to the other just decreases the probability of the favorable action, but the favorable action is the same as before. More details are in Appendix C.4.

The desired configurations across the four scenarios share common properties: the number of allowed actions (K) is set to 2, 5, and 9; the inner VASLAs are L_{R-I} with $\lambda_1 = 0.01$ and $\lambda_2 = 0$; and the initial depths are 1, 3, 5, and 7 for configurations 1 through 4, respectively. All reported results are based on 10000 iterations.

C.1. Scenario 1

The aim of this scenario is to evaluate the proposed learning automaton in a Markovian switching environment with respect to TNR and TNAS. The considered environment is a two-state Markov chain in which switching from one state to the other changes the favorable action. In addition, the tendency to change from one state to the other is very low.

From a mathematical standpoint, in the transition matrix of the Markov chain, the probability of $a_{ij} \quad i = j$ is greater than $a_{ij} \quad i \neq j$. The following transition matrix is considered to reach our desired condition.

$$T = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \tag{C.1}$$

The following reward matrix is regarded to satisfy the condition of this experiment. Also, Fig. C.20.a visualizes the desired Markovian switching environment.

$$R = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \tag{C.2}$$

Before analyzing the results which are shown in Table C.14, it is better to convert the Markovian switching environment to the stationary environment with the steady-state analysis of the Markov chain as follows:

$$\begin{gathered} v_\infty T = v_\infty \\ v_\infty \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} = v_\infty \\ \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} = \begin{pmatrix} x & y \end{pmatrix} \\ \begin{cases} 0.9x + 0.1y = x \\ 0.1x + 0.9y = y \\ x + y = 1 \end{cases} \\ \begin{cases} x = 0.5 \\ y = 0.5 \end{cases} \end{gathered} \tag{C.3}$$

Multiplying the steady-state matrix by the reward probability matrix of the Markovian switching environment yields the reward probability matrix of the equivalent stationary environment.

$$\begin{gathered} v_\infty R_{Markovian \ \ Switching} = R_{Stationary} \\ \begin{pmatrix} 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix} \end{gathered} \tag{C.4}$$

Consequently, a learning automaton deals with a completely random environment in which both action 1 and action 2 have the same probability of being rewarded, equal to 0.5.

By examining the results (Table C.14) and the mathematical analysis together, we can understand that the designed environment is completely random, making the learning process in such an environment impossible. Most of the time, the AVDHLA outperforms the FSLA with respect to TNR and TNAS because of the flexibility of this learning automaton to choose the desired depth for each action separately. Overall, we cannot expect these automata to learn from this environment.

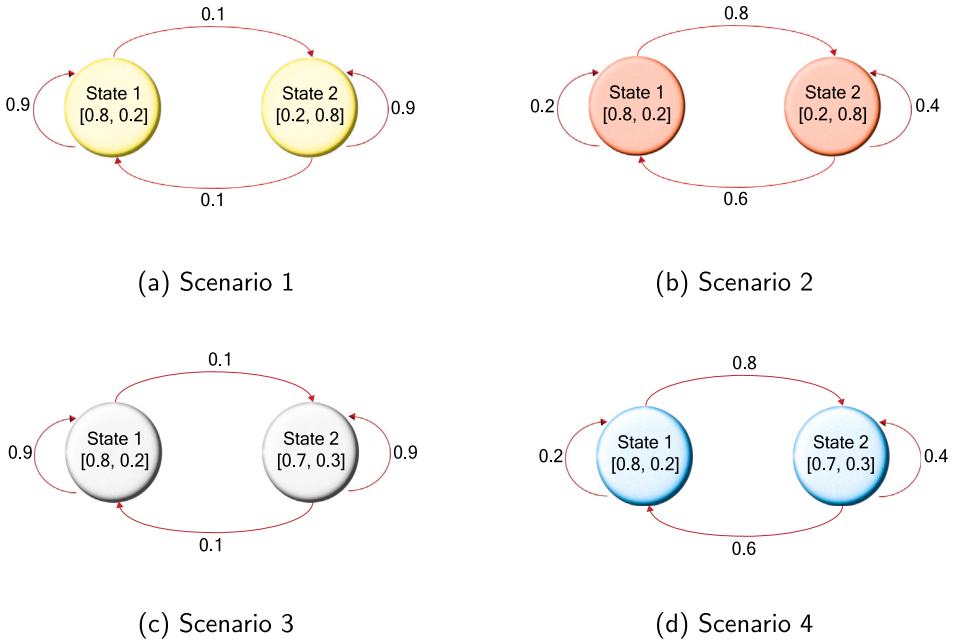


Fig. C.20. Four scenarios of Markovian switching environment.

Table C.14
Experimental results of the first scenario with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
				K = 2				
AVDHLA	**6189**	**1054**	**5638**	**376**	**5580**	**376**	**5829**	**480**
FSLA	6350	3650	6408	982	6093	667	5730	573
				K = 5				
AVDHLA	**4905**	**1682**	**5003**	**2973**	**4942**	**2190**	**4831**	**1951**
FSLA	4090	5910	5248	2654	5103	2073	5027	1729
				K = 9				
AVDHLA	**4741**	**3695**	**4226**	**2932**	**4295**	**2488**	**4245**	**2355**
FSLA	2764	7236	4300	3960	4320	3277	4202	2984

C.2. Scenario 2

This scenario is conducted to study the proposed learning automaton in a Markovian switching environment in which switching from one state to the other will completely change the favorable action. Additionally, the tendency to change from one state to the other is relatively high.

Mathematically, in the transition matrix of the Markov chain, the probability of $a_{ij} \quad i = j$ is less than $a_{ij} \quad i \neq j$. The following transition matrix is considered to reach our desired condition.

$$T = \begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix} \tag{C.5}$$

The following reward matrix is used to satisfy the conditions of this experiment. Also, Fig. C.20.b visualizes the desired Markovian switching environment.

$$R = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \tag{C.6}$$

Before analyzing the results shown in Table C.15, it is better to convert the Markovian switching environment to a stationary environment using the steady-state analysis of the Markov chain, as follows:

$$\begin{gathered} v_\infty T = v_\infty \\ v_\infty \begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix} = v_\infty \\ \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix} = \begin{pmatrix} x & y \end{pmatrix} \\ \begin{cases} 0.2x + 0.6y = x \\ 0.8x + 0.4y = y \\ x + y = 1 \end{cases} \\ \begin{cases} x = 0.42 \\ y = 0.58 \end{cases} \end{gathered} \tag{C.7}$$

Multiplying the steady-state matrix by the reward probability matrix of the Markovian switching environment yields the reward probability matrix of the equivalent stationary environment.

$$\begin{gathered} v_\infty R_{Markovian \quad Switching} = R_{Stationary} \\ \begin{pmatrix} 0.42 & 0.58 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} = \begin{pmatrix} 0.45 & 0.54 \end{pmatrix} \end{gathered} \tag{C.8}$$

Therefore, the second action is more likely to be rewarded by the surrounding environment. We should note that the difference between these automata is not large enough to say that one action is favorable, but we expect the automata to learn to choose the second action more than the first one.

The results (Table C.15) and mathematical analysis show that the AVDHLA outperforms the FSLA with respect to TNR and TNAS. This performance can be seen at lower initial depths. The underlying reason is the confusion of the learning automaton with changing the favorable action in each state and the tendency of the environment to change the current state. As a matter of fact, higher values for depth show bias towards the favorable action, and as a result, making the right decision for the learning automaton comes at a lot of cost. Overall, we

Table C.15
Experimental results of the second scenario with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
				K = 2				
AVDHLA	**5177**	**565**	**5183**	**946**	**5105**	**765**	**5362**	**93**
FSLA	4291	5709	4971	1570	5101	811	5218	494
				K = 5				
AVDHLA	**3834**	**2490**	**3635**	**3013**	**3761**	**2639**	**3772**	**2554**
FSLA	2014	7986	3134	4285	3436	3448	3663	2872
				K = 9				
AVDHLA	**2267**	**5636**	**2415**	**5292**	**2816**	**4493**	**2884**	**4331**
FSLA	1150	8850	2170	5973	2298	5526	2685	4722

Table C.16
Experimental results of the third scenario with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
				K = 2				
AVDHLA	**7551**	**33**	**7421**	**216**	**7422**	**90**	**7469**	**73**
FSLA	6319	3681	7346	361	7433	60	7424	7
				K = 5				
AVDHLA	**4566**	**2384**	**4690**	**2316**	**4650**	**2043**	**4696**	**2107**
FSLA	3604	6396	4677	2888	4465	2397	4609	2117
				K = 9				
AVDHLA	**4224**	**3630**	**4162**	**4071**	**4168**	**3413**	**4385**	**2553**
FSLA	2865	7135	4149	4149	4110	3585	4066	2900

can infer that in such environments where the favorable action changes periodically, the AVDHLA might be a good choice.

C.3. Scenario 3

The purpose of this scenario is to assess the proposed learning automaton in a Markovian switching environment in conjunction with TNR and TNAS. The considered environment is a two-state Markov chain in which switching from one state to the other will decrease or increase the probability of being rewarded in terms of the favorable action, but that action is still the favorable one. As well, the environment intends to stay in the current state.

Analytically, in the transition matrix of the Markov chain, the probability of element $a_{ij} \quad i = j$ is more significant than $a_{ij} \quad i \neq j$. The transition matrix of such an environment is as follows:

$$T = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \tag{C.9}$$

To reach the desired environment, we consider the following reward matrix for the Markov chain. Additionally, Fig. C.20.c shows the desired Markovian switching environment.

$$R = \begin{pmatrix} 0.8 & 0.2 \\ 0.7 & 0.3 \end{pmatrix} \tag{C.10}$$

Before analysis of the result in Table C.16, conversion of the Markovian switching environment to the stationary environment should be done using the steady state analysis of the Markov chain as follows:

$$\begin{gathered} v_\infty T = v_\infty \\ v_\infty \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} = v_\infty \\ \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} = \begin{pmatrix} x & y \end{pmatrix} \\ \begin{cases} 0.9x + 0.1y = x \\ 0.1x + 0.9y = 1 \\ x + y = 1 \end{cases} \\ \begin{cases} x = 0.5 \\ y = 0.5 \end{cases} \end{gathered} \tag{C.11}$$

Getting the reward matrix of the stationary environment depends on the multiplication of the steady state matrix to the reward probability matrix of the Markovian Switching environment.

$$\begin{gathered} v_\infty R_{Markovian \;\; Switching} = R_{Stationary} \\ \begin{pmatrix} 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.7 & 0.3 \end{pmatrix} = \begin{pmatrix} 0.75 & 0.25 \end{pmatrix} \end{gathered} \tag{C.12}$$

Accordingly, the Markovian switching environment is transformed into a stationary environment where action 1 is favored 75% of the time, and action 2 is favored 25% of the time.

Table C.17
Experimental results of the last scenario with respect to TNR and TNAS.

Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
				K = 2				
AVDHLA	**7323**	**129**	**7449**	**36**	**7409**	**23**	**7469**	**1**
FSLA	6148	3852	7227	398	7365	63	7380	6
				K = 5				
AVDHLA	**2807**	**4509**	**2997**	**4064**	**2784**	**4775**	**3028**	**4050**
FSLA	1997	8003	2674	4975	2946	4251	3001	3999
				K = 9				
AVDHLA	**2275**	**5740**	**2549**	**5135**	**2934**	**4225**	**2742**	**4538**
FSLA	1231	8769	2219	5850	2522	5012	2613	4855

The results (Table C.16) show the superiority of the AVDHLA over the FSLA with respect to TNR and TNAS. This superiority is more obvious in lower values of K and N. The main reason for the first one is that the lower value of K can help the AVDHLA to tune the depth of each action faster. Consequently, it performs better by getting more rewards with a lower number of action switching. The second superiority is in low depth since the AVDHLA can asymmetrically choose its depth relative to the FSLA. As expected, this learning automaton works better in such an environment in conjunction with the experiment metrics.

C.4. Scenario 4

This scenario is conducted to evaluate the proposed learning automaton in a Markovian switching environment in conjunction with TNR and TNAS. The environment is modeled as a two-state Markov chain in which the favorable action in all states is the same as before, but changing the state can decrease or increase the probability of being rewarded for the favorable action. Additionally, the environment tries to change the current state.

Mathematically, in the transition matrix of the Markov chain, the probability of $a_{ij} \quad i = j$ is less than $a_{ij} \quad i \neq j$. The following transition matrix is considered for this scenario.

$$T = \begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix} \tag{C.13}$$

The corresponding reward matrix that satisfies the specified conditions is as follows. Additionally, in Fig. C.20.d, the desired Markovian switching environment is depicted.

$$R = \begin{pmatrix} 0.8 & 0.2 \\ 0.7 & 0.3 \end{pmatrix} \tag{C.14}$$

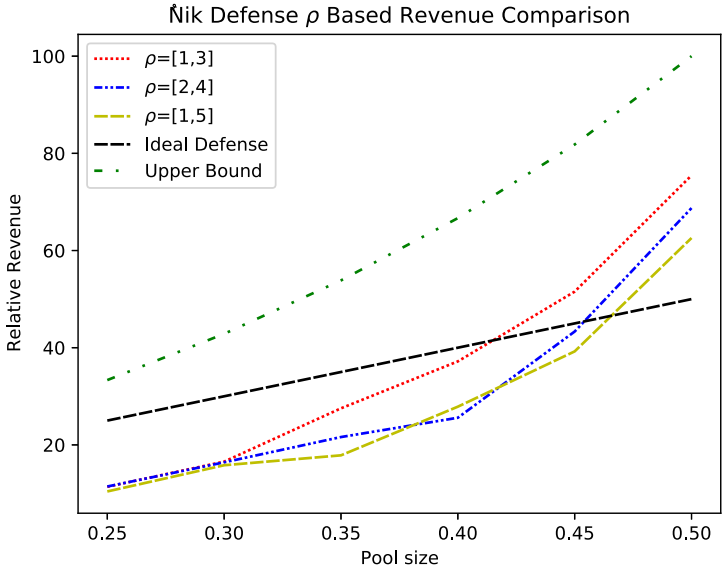


Fig. D.21. Experimental results of Scenario 1 with respect to the relative revenue.

To have a better overview of the specified environment, we need to convert the Markovian switching environment into the stationary environment using the steady-state analysis of the Markov chain as follows:

$$\begin{gathered} v_\infty T = v_\infty \\ v_\infty \begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix} = v_\infty \\ \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix} = \begin{pmatrix} x & y \end{pmatrix} \\ \begin{cases} 0.2x + 0.6y = x \\ 0.8x + 0.4y = 1 \\ x + y = 1 \end{cases} \\ \begin{cases} x = 0.42 \\ y = 0.58 \end{cases} \end{gathered} \tag{C.15}$$

The final result is obtained by multiplying the steady-state matrix with the reward probability matrix of the Markovian switching environment.

$$\begin{gathered} v_\infty R_{Markovian\ \ Switching} = R_{Stationary} \\ \begin{pmatrix} 0.42 & 0.58 \end{pmatrix} \begin{pmatrix} 0.8 & 0.2 \\ 0.7 & 0.3 \end{pmatrix} = \begin{pmatrix} 0.74 & 0.26 \end{pmatrix} \end{gathered} \tag{C.16}$$

Consequently, the Markovian switching environment turns into a stationary environment, where the probability of being rewarded for the first action is 0.74 and for the second action is 0.26.

By looking at the results (Table C.17), the AVDHLA outperforms the FSLA with respect to TNR and TNAS. In such an environment with the existence of the favorable action, the AVDHLA can increase its depth through the favorable action. As a result, it will collect more rewards with less action switching. Eventually, the AVDHLA can be robust in this environment in conjunction with the considered metrics.

Appendix D. Nik defense experiment extras

This appendix is dedicated to performing additional experiments on the proposed defense with the AVDHLA as a decision-maker. Three scenarios are considered as follows:

1. This scenario examines the effect of the fail-safe parameter on the quality of the proposed defense.
2. To understand how τ parameter affects the quality of the proposed defense, this scenario is conducted.
3. As well as other parameters, the number of τ in one θ will be studied to see the quality of the proposed defense with the various values of θ.

We should remark that 10000 blocks are generated for each scenario. Inner VASLAs are $L_{R-\epsilon P}$ with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$.

D.1. Scenario 1

This scenario is designed to examine the proposed defense under various values of ρ to evaluate the impact of the fail-safe parameter on the relative revenue and the lower bound threshold. For this purpose, intervals of ρ, such as $[1,3]$, $[2,4]$, and $[1,5]$, are considered for the experiment. The values of ρ can vary discretely between the minimum and maximum. The value of τ corresponds to the time required to mine five blocks, and each θ is divided into ten time intervals.

The results in Fig. D.21 demonstrate the superiority of $[1,5]$ for the AVDHLA with respect to the relative revenue. It shows the tendency of the learning automaton towards the higher values of ρ. In the AVDHLA, the learning automaton asymmetrically chose the depth for its action, so longer intervals can help it to choose the accurate value for the depth.

On the other hand, observations reveal similar results for the lower bound threshold. Overall, we can conclude that longer fail-safe intervals are more advantageous for the AVDHLA due to the increased flexibility to choose asymmetrically in relation to the experimental metrics.

D.2. Scenario 2

The ultimate objective of this scenario is to study the impact of the τ time interval on the quality of the proposed defense with respect to the relative revenue. To achieve this, three different values of τ are considered: $\tau = 5$, 9, and 18. The ρ parameter swings between $\rho_{min} = 1$ and $\rho_{max} = 3$. For the sake of example, $\tau = 5$ means that one τ time interval is equivalent to the mining time of five blocks. Also, one θ has ten τ intervals.

The obtained results are displayed in Fig. D.22. These results show that lower values of τ result in reduced relative revenue for selfish miners. This aligns with our expectations, as higher values of τ allow for a more thorough examination of the created forks. Consequently, the learning automaton receives more accurate feedback from the environment, enabling it to tune the β parameter more effectively.

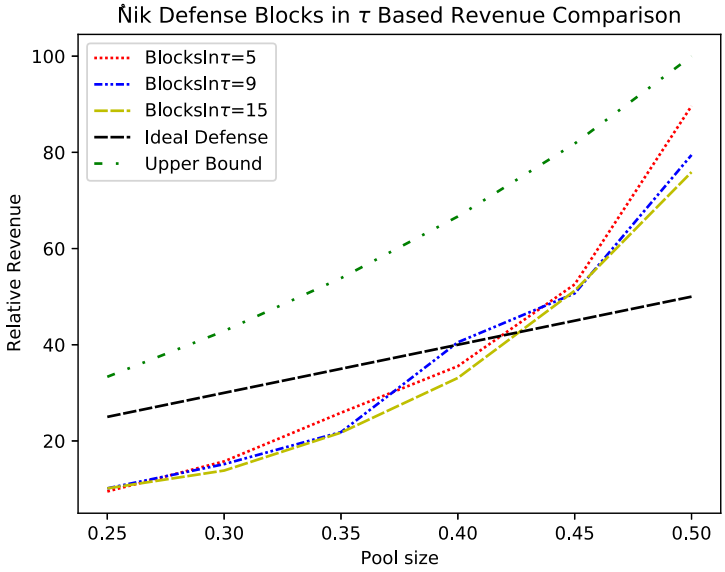


Fig. D.22. Experimental results of Scenario 2 with respect to the relative revenue.

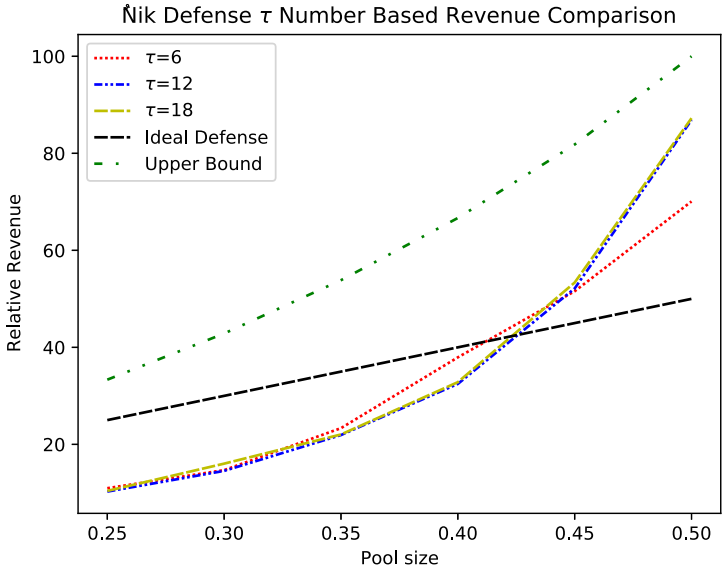


Fig. D.23. Experimental results of Scenario 3 with respect to the relative revenue.

D.3. Scenario 3

The last scenario is about investigating the impact of the number of τ in one θ on the performance of the proposed defense in relation to the relative revenue. For this purpose, τ is considered to be about the time it takes to mine five blocks. Additionally, three values of 6, 12, and 18 are used for the number of τ in one θ.

By looking at the results in Fig. D.23 for the AVDHLA, the superiority of the lower number of τ in one θ with respect to the relative revenue of the selfish miners is obvious.

As predicted from the behavior of the proposed learning automaton, the less number of τ in one θ leads to more number of θ in one simulation. This means that the learning automaton can adapt itself to a complex environment such as blockchain by setting the β parameter.

Eventually, we can claim from this scenario of the experiment and the previous one (Appendix D.2) that tuning the reinforcement signal accurately leads to a more expensive selfish mining attack. As a result, we have a better defense in conjunction with the relative revenue.

Data availability

No data was used for the research described in the article.

References

[1] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[2] A. Alharin, T.-N. Doan, M. Sartipi, Reinforcement learning interpretation methods: A survey, IEEE Access 8 (2020) 171058–171077.
[3] H.-n. Wang, N. Liu, Y.-y. Zhang, D.-w. Feng, F. Huang, D.-s. Li, Y.-m. Zhang, Deep reinforcement learning: a survey, Front. Inf. Technol. Electron. Eng. 21 (12) (2020) 1726–1744.
[4] S. Gronauer, K. Diepold, Multi-agent deep reinforcement learning: a survey, Artif. Intell. Rev. (2022) 1–49.
[5] R.T. Akella, B. Eysenbach, J. Schneider, R. Salakhutdinov, Distributional distance classifiers for goal-conditioned reinforcement learning, in: ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems, 2023.
[6] B. Eysenbach, J. Tyo, S. Gu, G. Brain, R. Salakhutdinov, Z. Lipton, S. Levine, Reinforcement learning with unknown reward functions, in: Task-Agnostic Reinforcement Learning Workshop at ICLR 2019, 2019.

[7] B. Eysenbach, R.R. Salakhutdinov, S. Levine, Search on the replay buffer: Bridging planning and reinforcement learning, Adv. Neural Inf. Process. Syst. 32 (2019).

[8] C. Zheng, B. Eysenbach, H. Walke, P. Yin, K. Fang, R. Salakhutdinov, S. Levine, Stabilizing contrastive RL: Techniques for offline goal reaching, 2023, arXiv preprint arXiv:2306.03346.

[9] M. Chen, A. Pacchiano, X. Zhang, State-free reinforcement learning, arXiv preprint arXiv:2409.18439 (2024).

[10] H. Yuan, C. Ni, H. Wang, X. Zhang, L. Cong, C. Szepesvári, M. Wang, Bandit theory and thompson sampling-guided directed evolution for sequence optimization, Adv. Neural Inf. Process. Syst. 35 (2022) 38291–38304.

[11] S. Yang, X. Zhang, M. Wang, Decentralized gossip-based stochastic bilevel optimization over communication networks, Adv. Neural Inf. Process. Syst. 35 (2022) 238–252.

[12] C. Zhao, R. Yang, B. Wang, X. Zhang, S. Li, Learning adversarial low-rank markov decision processes with unknown transition and full-information feedback, Adv. Neural Inf. Process. Syst. 36 (2024).

[13] M.A. Thathachar, P.S. Sastry, Varieties of learning automata: an overview, IEEE Trans. Syst. Man Cybern. B 32 (6) (2002) 711–722.

[14] K.S. Narendra, M.A. Thathachar, Learning Automata: An Introduction, Courier corporation, 2012.

[15] L. Jiao, X. Zhang, O.-C. Granmo, K.D. Abeyrathna, On the convergence of tsetlin machines for the XOR operator, IEEE Trans. Pattern Anal. Mach. Intell. 45 (5) (2022) 6072–6085.

[16] X. Zhang, L. Jiao, O.-C. Granmo, M. Goodwin, On the convergence of tsetlin machines for the IDENTITY-and NOT operators, IEEE Trans. Pattern Anal. Mach. Intell. 44 (10) (2021) 6345–6359.

[17] D. Abeyrathna, O.-C. Granmo, M. Goodwin, Convolutional regression tsetlin machine: An interpretable approach to convolutional regression, in: 2021 6th International Conference on Machine Learning Technologies, 2021, pp. 65–73.

[18] J. Sharma, O.-C. Granmo, L. Jiao, On the equivalence of the weighted Tsetlin machine and the perceptron, 2022, arXiv preprint arXiv:2212.13634.

[19] H. Guo, S. Wang, J. Fan, S. Li, Learning automata based incremental learning method for deep neural networks, IEEE Access 7 (2019) 41164–41171.

[20] D.L. Barabási, T. Beynon, Á. Katona, N. Perez-Nieves, Complex computation from developmental priors, Nature Commun. 14 (1) (2023) 2226.

[21] B. Bhattarai, O.-C. Granmo, L. Jiao, ConvTextTM: An explainable convolutional Tsetlin machine framework for text classification, in: Proceedings of the Thirteenth Language Resources and Evaluation Conference, 2022, pp. 3761–3770.

[22] B. Bhattarai, O.-C. Granmo, L. Jiao, R. Yadav, J. Sharma, Tsetlin machine embedding: Representing words using logical expressions, 2023, arXiv preprint arXiv:2301.00709.

[23] A.B. Hashemi, M.R. Meybodi, A note on the learning automata based algorithms for adaptive parameter selection in PSO, Appl. Soft Comput. 11 (1) (2011) 689–705.

[24] K.S. Narendra, K. Parthasarathy, Learning automata approach to hierarchical multiobjective analysis, IEEE Trans. Syst. Man Cybern. 21 (1) (1991) 263–272.

[25] R. Ayanzadeh, M. Halem, T. Finin, Reinforcement quantum annealing: A hybrid quantum learning automata, Sci. Rep. 10 (1) (2020) 7952.

[26] A. Reina, Robot teams stay safe with blockchains, Nat. Mach. Intell. 2 (5) (2020) 240–241.

[27] A. Yazidi, D. Silvestre, B.J. Oommen, Solving two-person zero-sum stochastic games with incomplete information using learning automata with artificial barriers, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[28] B.J. Oommen, R.O. Omslandseter, L. Jiao, Learning automata-based partitioning algorithms for stochastic grouping problems with non-equal partition sizes, Pattern Anal. Appl. 26 (2) (2023) 751–772.

[29] R.W. Thomas, D.H. Friend, L.A. DaSilva, A.B. MacKenzie, Cognitive networks: adaptation and learning to achieve end-to-end performance objectives, IEEE Commun. Mag. 44 (12) (2006) 51–57.

[30] A. Yazidi, I. Hassan, H.L. Hammer, B.J. Oommen, Achieving fair load balancing by invoking a learning automata-based two-time-scale separation paradigm, IEEE Trans. Neural Netw. Learn. Syst. 32 (8) (2020) 3444–3457.

[31] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, P. Sarigiannidis, M.S. Obaidat, Adaptive wireless networks using learning automata, IEEE Wirel. Commun. 18 (2) (2011) 75–81.

[32] A. Rezvanian, S.M. Vahidipour, M.R. Meybodi, A new stochastic diffusion model for influence maximization in social networks, Sci. Rep. 13 (1) (2023) 6122.

[33] M.M.D. Khomami, M.R. Meybodi, A. Rezvanian, Exploring social networks through stochastic multilayer graph modeling, Chaos Solitons Fractals 182 (2024) 114764.

[34] M.M.D. Khomami, M.R. Meybodi, A. Rezvanian, Efficient identification of maximum independent sets in stochastic multilayer graphs with learning automata, Results Eng. (2024) 103224.

[35] M.M.D. Khomami, A. Rezvanian, M.R. Meybodi, A. Bagheri, Cfin: a community-based algorithm for finding influential nodes in complex social networks, J. Supercomput. 77 (3) (2021) 2207–2236.

[36] Z. Zhang, D. Wang, J. Gao, Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks, IEEE Trans. Neural Netw. Learn. Syst. 32 (10) (2020) 4639–4652.

[37] X. Hou, L. Chen, J. Tang, J. Li, Multi-agent learning automata for online adaptive control of large-scale traffic signal systems, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 1497–1502.

[38] X. Fang, J. Wang, C. Yin, Y. Han, Q. Zhao, Multiagent reinforcement learning with learning automata for microgrid energy management and decision optimization, in: 2020 Chinese Control and Decision Conference, CCDC, IEEE, 2020, pp. 779–784.

[39] M.A. Thathachar, P.S. Sastry, Networks of Learning Automata: Techniques for Online Stochastic Optimization, Springer Science & Business Media, 2003.

[40] M. Thathachar, P.S. Sastry, A new approach to the design of reinforcement schemes for learning automata, IEEE Trans. Syst. Man Cybern. (1) (1985) 168–175.

[41] G.I. Papadimitriou, M. Sklira, A.S. Pomportsis, A new class of/spl epsi/-optimal learning automata, IEEE Trans. Syst. Man Cybern. B 34 (1) (2004) 246–254.

[42] M. Thathachar, B.R. Harita, Learning automata with changing number of actions, IEEE Trans. Syst. Man Cybern. 17 (6) (1987) 1095–1100.

[43] O.-C. Granmo, The Tsetlin machine–A game theoretic bandit driven approach to optimal pattern recognition with propositional logic, 2018, arXiv preprint arXiv:1804.01508.

[44] S. Leonardos, G. Piliouras, K. Spendlove, Exploration-exploitation in multi-agent competition: convergence with bounded rationality, Adv. Neural Inf. Process. Syst. 34 (2021) 26318–26331.

[45] I.J. Sledge, J.C. Príncipe, Balancing exploration and exploitation in reinforcement learning using a value of information criterion, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2017, pp. 2816–2820.

[46] Q. Chen, Q. Zhang, Y. Liu, Balancing exploration and exploitation in episodic reinforcement learning, Expert Syst. Appl. 231 (2023) 120801.

[47] X. He, K. Zhao, X. Chu, AutoML: A survey of the state-of-the-art, Knowl.-Based Syst. 212 (2021) 106622.

[48] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, Adv. Neural Inf. Process. Syst. 28 (2015).

[49] J.M. Kübler, V. Stimper, S. Buchholz, K. Muandet, B. Schölkopf, AutoML two-sample test, Adv. Neural Inf. Process. Syst. 35 (2022) 15929–15941.

[50] K.D. Abeyrathna, O.-C. Granmo, R. Shafik, A. Yakovlev, A. Wheeldon, J. Lei, M. Goodwin, A novel multi-step finite-state automaton for arbitrarily deterministic tsetlin machine learning, in: International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2020, pp. 108–122.

[51] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Decentralized Bus. Rev. (2008).

[52] I. Eyal, E.G. Sirer, Majority is not enough: Bitcoin mining is vulnerable, Commun. ACM 61 (7) (2018) 95–102.

[53] A. Nikhalat-Jahromi, A.M. Saghiri, M.R. Meybodi, Nik defense: An artificial intelligence based defense mechanism against selfish mining in Bitcoin, 2023, arXiv e-prints, arXiv–2301.

[54] T.M. Moerland, J. Broekens, A. Plaat, C.M. Jonker, et al., Model-based reinforcement learning: A survey, Found. Trends® Mach. Learn. 16 (1) (2023) 1–118.

[55] F. Yi, W. Fu, H. Liang, Model-based reinforcement learning: A survey, 2018.

[56] S. Çalışır, M.K. Pehlivanoğlu, Model-free reinforcement learning algorithms: A survey, in: 2019 27th Signal Processing and Communications Applications Conference, SIU, IEEE, 2019, pp. 1–4.

[57] J. Ramírez, W. Yu, A. Perrusquía, Model-free reinforcement learning from expert demonstrations: a survey, Artif. Intell. Rev. (2022) 1–29.

[58] Y. Liu, A. Halev, X. Liu, Policy learning with constraints in model-free reinforcement learning: A survey, in: The 30th International Joint Conference on Artificial Intelligence, IJCAI, 2021.

[59] A. Rezvanian, A.M. Saghiri, S.M. Vahidipour, M. Esnaashari, M.R. Meybodi, Recent Advances in Learning Automata, vol. 754, Springer, 2018.

[60] J. Zhang, M. Zhou, Learning Automata and Their Applications to Intelligent Systems, John Wiley & Sons, 2023.

[61] R. Seraj, J. Sharma, O.-C. Granmo, Tsetlin machine for solving contextual bandit problems, Adv. Neural Inf. Process. Syst. 35 (2022) 30194–30205.

[62] K.D. Abeyrathna, B. Bhattarai, M. Goodwin, S.R. Gorji, O.-C. Granmo, L. Jiao, R. Saha, R.K. Yadav, Massively parallel and asynchronous tsetlin machine architecture supporting almost constant-time scaling, in: International Conference on Machine Learning, PMLR, 2021, pp. 10–20.

[63] M.R. Meybodi, H. Beigy, A note on learning automata-based schemes for adaptation of BP parameters, Neurocomputing 48 (1–4) (2002) 957–974.

[64] A. Jamalian, S. Mehrabi, Emotional learning automaton, in: 2022 IEEE 21st International Conference on Cognitive Informatics & Cognitive Computing, ICCI* CC, IEEE, 2022, pp. 99–105.

[65] M.R. Khojasteh, M.R. Meybodi, Using learning automata in cooperation among agents in a team, in: 2005 Portuguese Conference on Artificial Intelligence, IEEE, 2005, pp. 306–312.

[66] M.L. Tsetlin, On behaviour of finite automata in random medium, Avtomat. i Telemekh. 22 (10) (1961) 1345–1354.

[67] M. Ghalavand, J. Hatami, S.K. Setarehdan, F. Nosrati, H. Ghalavand, A. Nikhalat-Jahromi, Comparison of the effects of interaction with intentional agent and artificial intelligence using fNIRS, arXiv preprint arXiv:2402.17650 (2024).

[68] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C.W. Omlin, G.T. Berge, The convolutional Tsetlin machine, 2019, arXiv preprint arXiv:1905.09688.

[69] J. Sharma, R. Yadav, O.-C. Granmo, L. Jiao, Drop clause: Enhancing performance, robustness and pattern recognition capabilities of the tsetlin machine, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 13547–13555.

[70] K. Darshana Abeyrathna, O.-C. Granmo, X. Zhang, L. Jiao, M. Goodwin, The regression Tsetlin machine: a novel approach to interpretable nonlinear regression, Phil. Trans. R. Soc. A 378 (2164) (2020) 20190165.

[71] A. Phoulady, O.-C. Granmo, S.R. Gorji, H.A. Phoulady, The weighted tsetlin machine: compressed representations with weighted clauses, 2019, arXiv preprint arXiv:1911.12607.

[72] K.D. Abeyrathna, O.-C. Granmo, M. Goodwin, Extending the tsetlin machine with integer-weighted clauses for increased interpretability, IEEE Access 9 (2021) 8233–8248.

[73] S. Glimsdal, O.-C. Granmo, Coalesced multi-output tsetlin machines with clause sharing, 2021, arXiv preprint arXiv:2108.07594.

[74] M. Khaksar Manshad, M.R. Meybodi, A. Salajegheh, A variable action set cellular learning automata-based algorithm for link prediction in online social networks, J. Supercomput. 77 (7) (2021) 7620–7648.

[75] N. Fatehi, H.S. Shahhoseini, J. Wei, C.-T. Chang, An automata algorithm for generating trusted graphs in online social networks, Appl. Soft Comput. 118 (2022) 108475.

[76] R. Ebrahim Pourian, M. Fartash, J. Akbari Torkestani, A deep learning model for energy-aware task scheduling algorithm based on learning automata for fog computing, Comput. J. 67 (2) (2024) 508–518.

[77] S. Ghanavati, J. Abawajy, D. Izadi, Automata-based dynamic fault tolerant task scheduling approach in fog computing, IEEE Trans. Emerg. Top. Comput. 10 (1) (2020) 488–499.

[78] M.G. Farahani, J.A. Torkestani, M. Rahmani, Adaptive personalized recommender system using learning automata and items clustering, Inf. Syst. 106 (2022) 101978.

[79] F. Safara, A. Souri, S.F. Deiman, Super peer selection strategy in peer-to-peer networks based on learning automata, Int. J. Commun. Syst. 33 (6) (2020) e4296.

[80] A. Yazidi, I. Hassan, H.L. Hammer, B.J. Oommen, Achieving fair load balancing by invoking a learning automata-based two-time-scale separation paradigm, IEEE Trans. Neural Netw. Learn. Syst. 32 (8) (2020) 3444–3457.

[81] A. Yazidi, D. Silvestre, B.J. Oommen, Solving two-person zero-sum stochastic games with incomplete information using learning automata with artificial barriers, IEEE Trans. Neural Netw. Learn. Syst. 34 (2) (2021) 650–661.

[82] C. Dey, R. Bose, K.K. Ghosh, S. Malakar, R. Sarkar, LAGOA: Learning automata based grasshopper optimization algorithm for feature selection in disease datasets, J. Ambient Intell. Humaniz. Comput. (2022) 1–20.

[83] S. Gholami, A.M. Saghiri, S. Vahidipour, M. Meybodi, HLA: a novel hybrid model based on fixed structure and variable structure learning automata, J. Exp. Theor. Artif. Intell. 35 (2) (2023) 231–256.

[84] A. Nikhalat-Jahromi., A. Saghiri., M. Meybodi., Q-defense: When Q-learning comes to help proof-of-work against the selfish mining attack, in: Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART, INSTICC, SciTePress, 2024, pp. 37–46, http://dx.doi.org/10.5220/0012378600003636.

[85] A. Nikhalat-Jahromi, A.M. Saghiri, M.R. Meybodi, VDHLA: Variable depth hybrid learning automaton and its application to defense against the selfish mining attack in Bitcoin, 2023, arXiv preprint arXiv:2302.12096.

[86] I. Eyal, The miner's dilemma, in: 2015 IEEE Symposium on Security and Privacy, IEEE, 2015, pp. 89–103.

[87] K. Nayak, S. Kumar, A. Miller, E. Shi, Stubborn mining: Generalizing selfish mining and combining with an eclipse attack, in: 2016 IEEE European Symposium on Security and Privacy, EuroS&P, IEEE, 2016, pp. 305–320.

[88] W. Wang, D.T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, D.I. Kim, A survey on consensus mechanisms and mining strategy management in blockchain networks, IEEE Access 7 (2019) 22328–22370.

[89] M. Babaioff, S. Dobzinski, S. Oren, A. Zohar, On bitcoin and red balloons, in: Proceedings of the 13th ACM Conference on Electronic Commerce, 2012, pp. 56–73.

[90] T. Wang, S.C. Liew, S. Zhang, When blockchain meets AI: Optimal mining strategy achieved by machine learning, Int. J. Intell. Syst. 36 (5) (2021) 2183–2207.

[91] R. Zhang, B. Preneel, Publish or perish: A backward-compatible defense against selfish mining in bitcoin, in: Topics in Cryptology–CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14–17, 2017, Proceedings, Springer, 2017, pp. 277–292.

[92] S.S. Khanal, P. Prasad, A. Alsadoon, A. Maag, A systematic review: machine learning based recommendation systems for e-learning, Educ. Inf. Technol. 25 (4) (2020) 2635–2664.

[93] H. Ko, S. Lee, Y. Park, A. Choi, A survey of recommendation systems: recommendation models, techniques, and application fields, Electronics 11 (1) (2022) 141.

[94] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, R. Kashef, Recommendation systems: Algorithms, challenges, metrics, and business opportunities, Appl. Sci. 10 (21) (2020) 7748.

[95] Z. Cui, X. Xu, X. Fei, X. Cai, Y. Cao, W. Zhang, J. Chen, Personalized recommendation system based on collaborative filtering for IoT scenarios, IEEE Trans. Serv. Comput. 13 (4) (2020) 685–695.

[96] A. Da'u, N. Salim, Recommendation system based on deep learning methods: a systematic review and new directions, Artif. Intell. Rev. 53 (4) (2020) 2709–2748.

[97] A. Shankar, P. Perumal, M. Subramanian, N. Ramu, D. Natesan, V.R. Kulkarni, T. Stephan, An intelligent recommendation system in e-commerce using ensemble learning, Multimedia Tools Appl. 83 (16) (2024) 48521–48537.

[98] S. Dutta, S. Mondal, D. Sarkar, Design and implementation of recommendation system using sentiment analysis in social media, in: Proceedings of the International Conference on Computational Intelligence and Sustainable Technologies: ICoCIST 2021, Springer, 2022, pp. 141–152.

[99] U. Javed, K. Shaukat, I.A. Hameed, F. Iqbal, T.M. Alam, S. Luo, A review of content-based and context-based recommendation systems, Int. J. Emerg. Technol. Learn. (iJET) 16 (3) (2021) 274–306.

[100] S. Kadıoğlu, B. Kleynhans, Building higher-order abstractions from the components of recommender systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 22998–23004.

[101] E. Strong, B. Kleynhans, S. Kadıoğlu, MABWiser: parallelizable contextual multi-armed bandits, Int. J. Artif. Intell. Tools 30 (04) (2021) 2150021.

[102] E. Strong, B. Kleynhans, S. Kadioğlu, Mabwiser: A parallelizable contextual multi-armed bandit library for python, in: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence, ICTAI, IEEE, 2019, pp. 909–914.

[103] M. Thielbar, S. Kadıoğlu, C. Zhang, R. Pack, L. Dannull, Surrogate membership for inferred metrics in fairness evaluation, in: International Conference on Learning and Intelligent Optimization, Springer, 2023, pp. 424–442.

[104] F. Michalskỳ, S. Kadıoğlu, Surrogate ground truth generation to enhance binary fairness evaluation in uplift modeling, in: 2021 20th IEEE International Conference on Machine Learning and Applications, ICMLA, IEEE, 2021, pp. 1654–1659.

[105] Y. Jiang, S. Wu, H. Yang, H. Luo, Z. Chen, S. Yin, O. Kaynak, Secure data transmission and trustworthiness judgement approaches against cyber-physical attacks in an integrated data-driven framework, IEEE Trans. Syst. Man Cybern.: Syst. 52 (12) (2022) 7799–7809.

[106] B. Eysenbach, T. Zhang, S. Levine, R.R. Salakhutdinov, Contrastive learning as goal-conditioned reinforcement learning, Adv. Neural Inf. Process. Syst. 35 (2022) 35603–35620.

[107] Q. Su, F. Wang, D. Chen, G. Chen, C. Li, L. Wei, Deep convolutional neural networks with ensemble learning and transfer learning for automated detection of gastrointestinal diseases, Comput. Biol. Med. 150 (2022) 106054.

[108] J. Zhou, Z. Wu, Z. Jiang, K. Huang, K. Guo, S. Zhao, Background selection schema on deep learning-based classification of dermatological disease, Comput. Biol. Med. 149 (2022) 105966.

[109] B. Eysenbach, M. Geist, S. Levine, R. Salakhutdinov, A connection between one-step RL and critic regularization in reinforcement learning, in: International Conference on Machine Learning, PMLR, 2023, pp. 9485–9507.

[110] J. Tu, H. Chen, M. Wang, A.H. Gandomi, The colony predation algorithm, J. Bionic Eng. 18 (2021) 674–710.

[111] Y. Wang, T. Bai, T. Li, L. Huang, Osteoporotic vertebral fracture classification in X-rays based on a multi-modal semantic consistency network, J. Bionic Eng. 19 (6) (2022) 1816–1829.