Asymmetric Variable Depth Learning Automaton and Its Application in Defending Against Selfish Mining Attacks on Bitcoin

Ali Nikhalat-Jahromi[a], Ali Mohammad Saghiri[a], Mohammad Reza Meybodi[a]

[a]*Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran*

Abstract

Learning Automaton (LA) theory, 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 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 LA (AVDHLA). The AVDHLA model intelligently learns the depth of fixed structure LA in an autonomous manner by combining $L_{KN,K}$ from FSLA class and Variable Action Set LA (VASLA) from 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, AVDHLA consistently outperforms $L_{KN,K}$ 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.

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

1. Introduction

A learning automaton (LA) is a self-adaptive reinforcement learning [1, 2, 3, 4, 5, 6, 7, 8] model that enhances its performance through continuous interaction with a random environment. It learns to select the optimal action based on the feedback (response) which can be favorable or unfavorable it receives from the environment [9].

The inception of learning automaton can be traced back to Tsetlin's pioneering work [10], 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 [11, 12]; 2) Neural Networks: Convolutional Regression, similarity between perceptrons and Tsetlin, and deep neural networks [13, 14, 15, 16]; 3) NLP: Pattern recognition tasks using propositional logic and semantic representation of words [17, 18]; 4) Optimization Problems: Particle swarm and multilevel optimization [19, 20, 21, 22]; 5) Graph Theory: Partitioning problem [23, 24]; 6) Computer Networks: Cognitive radio, load balancing, and wireless networks [25, 26, 27]; 7) Social networks : Influence maximization [28] 8) Multi-agent Systems [29, 30, 31].

Our study focuses on two fundamental categories of learning automata, namely Fixed Structure Learning Automaton (FSLA) and Variable Structure Learning Automaton (VSLA) [10, 32, 33, 34]. FSLA belongs to the category of state machines, while VSLA involves the selection of actions from a finite set. VSLA updates its strategy based on a probability vector. For this research, we specifically chose $L_{KN,K}$ 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) [35], this type of LA serves as a key component in our novel LA framework.

$L_{KN,K}$ learning automata are generally used in online learning decision-making contexts, making offline depth adjustments impractical [10, 36]. 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 $L_{KN,K}$ to efficiently explore and exploit the environment [37, 38, 39]. 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 modification, increasing 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 $L_{KN,K}$ to optimize its performance [10].

Furthermore, from the perspective of Automated Machine Learning (AutoML) [40, 41, 42], $L_{KN,K}$ lacks internal mechanisms or tools to determine whether the selected depth for each action is appropriate or requires adjustment based on environmental conditions [10, 36, 43]. Therefore, an equal number of learning agents as the number of actions in $L_{KN,K}$ should be integrated with it. These auxiliary agents can aid $L_{KN,K}$ 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 $L_{KN,K}$ remains a significant challenge without current solutions. Our study introduces a novel solution to this critical problem. We propose the Asymmetric Variable Depth Hybrid LA (AVDHLA), a hybrid learning model that combines fixed structure and variable action set learning automata. AVDHLA effectively tackles the trade-off between explore and exploit by learning the appropriate depth, thus providing a solution to this issue.

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 $L_{KN,K}$. This parameter presents a challenge in effectively balancing the exploration and exploitation capabilities of $L_{KN,K}$.
2. We introduce a novel class of $L_{KN,K}$ that possesses the ability to learn its depth in a fully self-adaptive manner. The proposed model determines the appropriate depth by utilizing VASLA to learn from the performance of $L_{KN,K}$ 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 FSLA and VSLA.

4. We examine the effectiveness of the learning model in a trendy practical application: Bitcoin [44], a decentralized cryptocurrency. The main drawback of the Bitcoin consensus mechanism is that a type of attack named selfish mining might threaten its decentralization and fairness capabilities. This study uses AVDHLA to defend against the selfish mining attack [45, 46]. 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 $L_{KN,K}$. Analysis of the results shows the superiority of the proposed model in practical applications.
5. We showcase the practical versatility of 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 works 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, a pivotal paradigm in machine learning, can be categorized into two main types [1]: model-based [47, 48] and model-free [49, 50, 51] 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 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 chose learning automaton from model-free category.

The first class of the learning automaton family is fixed structure. The domain of fixed structure learning automaton [52, 53] is expansive, encompassing various branches such as $L_{KN,K}$ (Tsetlin) [36, 54, 55], Krinsky [56, 57], and

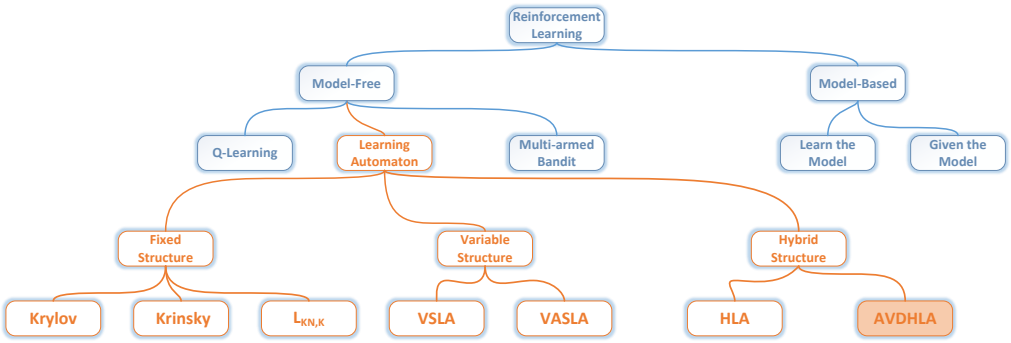


Figure 1: The taxonomy of reinforcement learning algorithms based on the model

Krylov [58]. 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 $L_{KN,K}$ [59] 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, this automaton encapsulates the intricate nature of human decision-making and calculation. Over its illustrious half-century existence, the $L_{KN,K}$ automaton [36, 60] has demonstrated its versatility and applicability across a wide spectrum of fields [10]. Notably, its recent integration with Neural Networks [61, 54, 55, 17, 62, 63, 64, 43, 65, 18] 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 [10]: (1) The clockwise policy for selecting the next action; (2) The fixed depth value.

Variable Structure Learning Automaton [10], belonging to the second class of learning automata, exhibits various types, with VSLA and VASLA being the most significant branches. In VSLA, interaction with the surrounding environment updates a probability vector. VASLA is similar to VSLA in its use of a probability vector. However, in certain situations, not all actions are available. Consequently, VASLA selects the next action based on the set of actions available at each stage.

It is noteworthy that both VSLA and VASLA have a wide range of applications, including: 1) Social Networks [66, 67]; 2) Fog Computing [68, 69]; 3) Recommender Systems [70]; 4) Computer Networks [71, 72]; 5) Optimizations [29, 73, 74]

Gholami et al. [75] introduced a significant modification to $L_{KN,K}$ [75] by combining the fixed structure and 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 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 [10, 32] 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 $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$. Subsequently, the environment evaluates the chosen action, emitting a reinforcement signal denoted as $\beta(t)$ from the set $\beta = \{\beta_1, \beta_2, ..., \beta_m\}$. This process is illustrated in Fig.2.

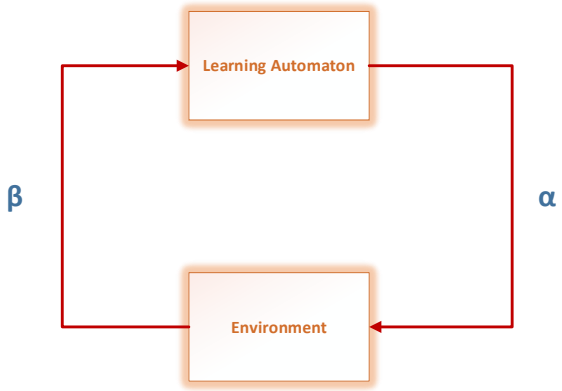


Figure 2: The interaction of LA and the surrounding environment

In the rest of this section, we will go deeper into an in-depth introduction

of FSLA, VSLA, and VASLA, which stand as key members within the learning automaton family.

3.1. Fixed Structure Learning Automaton (FSLA)

Fixed structure learning automaton [10, 32] 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 $L_{2N,2}$, which is a two-state $L_{KN,K}$. Afterward, $L_{2N,2}$ is expanded to obtain $L_{KN,K}$ as a general form of this family.

$L_{2,2}$ 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 $L_{2,2}$.

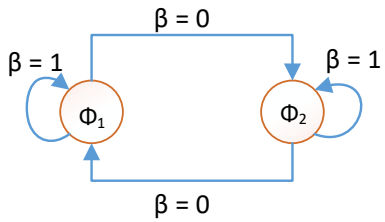


Figure 3: The $L_{2,2}$ learning automaton

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 $L_{2,2}$ to $L_{2N,2}$. $L_{2N,2}$ 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 $L_{2,2}$, 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. $L_{2N,2}$ learning automaton with $N = 2$ is shown in Fig.4.

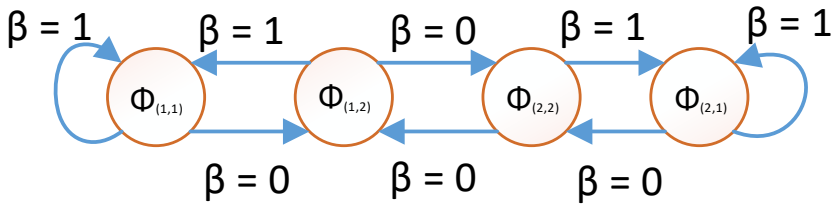


Figure 4: The $L_{2N,2}$ learning automaton with $N = 2$

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 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:

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

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, 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.

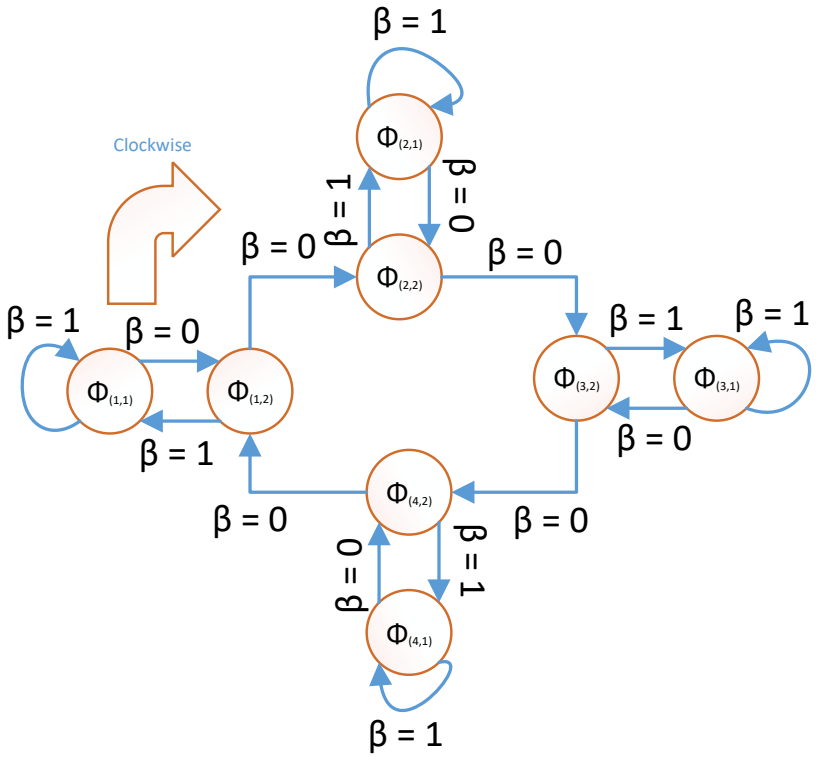


Figure 5: The $L_{KN,K}$ learning automaton with $K = 4$ and $N = 2$

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 [10, 32].

Variable structure can be defined mathematically by a quadruple $< \alpha, \beta, P, T >$, where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ denotes the finite action set from which the learning

automaton can select the intended action, $\beta = \{\beta_1, \beta_2, ..., \beta_k\}$ denotes the set of inputs (reinforcement signals) to the learning automaton, $P = \{p_1, p_2, ..., p_r\}$ 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 ($\beta = 0$):

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

In the above equations (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 with a monotonic rate.

- $\boldsymbol{L_{R-\varepsilon P}}$: 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 ($\lambda_1 >> \lambda_2$).

- $\boldsymbol{L_{R-I}}$: 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 doesn't 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 instant. To overcome this constraint, a subset of the variable structure learning automaton called the variable action set learning automaton [35] is defined. Like the variable structure, this automaton can be formulated by a quadruple $< \alpha, \beta, P, T >$, 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 $\{\alpha_1, \alpha_2, ..., \alpha_r\}$. 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:

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

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 instant 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:

$$p_i(t+1) = p_i(t) + \lambda_1 \beta (1 - p_i(t)) - \lambda_2 (1 - \beta) p_i(t) \tag{6}$$

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

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

4. Problem Formulation

In the classical $L_{KN,K}$ with K actions, each action i has N states denoted as $\phi_{(i,1)}$, $\phi_{(i,2)}$, ..., $\phi_{(i,N)}$ ($\phi_{(i,1)}$ 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 clock-wise 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 $L_{KN,K}$ in making decisions for both the next action and its depth.

To address the clockwise policy challenge, Gholami et al. [75] 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 $L_{KN,K}$ receives a penalty, the VSLA activates to select the action $i+1$ based on its historical experiences. This collaboration between VASLA and the $L_{KN,K}$ effectively resolves the clockwise policy issue in $L_{KN,K}$.

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 $i+1$, $VASLA_i$ 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 $\phi_{(i,2)}$, ..., $\phi_{(i,N)}, \phi_{(i,N+1)}$. Conversely, selecting the 'Shrink' action decreases the number of states, leading to $\phi_{(i,1)}$, ..., $\phi_{(i,N-2)}, \phi_{(i,N-1)}$. Choosing the 'Stop' action leaves the state unchanged. Over an extended duration, this collaborative interaction between VASLAs and $L_{KN,K}$ 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 section 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 Aymmetric 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.

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 Equation.1 (for unfavorable response) and Equation.7 (for favorable response).

To be more specific about the environment, we considered the P-model for modeling the outer environment of the proposed automaton in which the automaton may receive the reinforcement signal from the finite set of 0 or 1.

In the proposed automaton, consider the fixed structure learning automaton approaches the edge state. 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 FSLA will increase by one symmetrically.

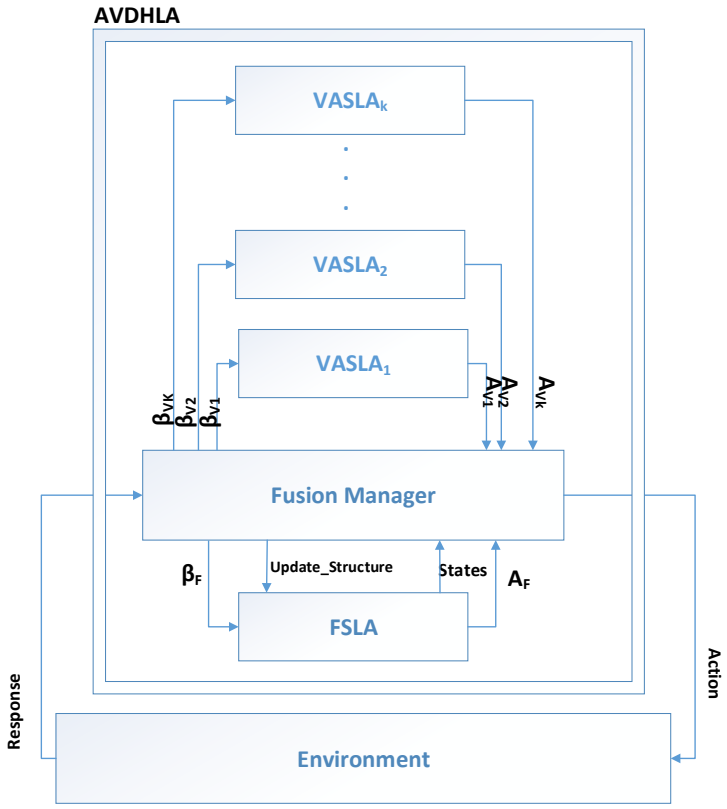


Figure 6: The architecture of AVDHLA

2. **Shrink**: If the $VASLA_i$ chooses this action, the depth of i^{th} action in FSLA will decrease by one symmetrically.
3. **Stop**: If the $VASLA_i$ chooses this action, the depth of i^{th} action in 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 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 $AVDHLA(K, N_1, ..., N_i, ..., N_K, \lambda_1, \lambda_2)$. 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 $L_{KN,K}$. 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 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 investigates the action switching. If the automaton penalizes from the outer environment and wants to change its action, this function returns true; otherwise, it will return false.
3. ***update_depth(new_depth_length)***: This function can update the depth of all actions in FSLA.

- ***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 βv_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 FSLA and 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 outer 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 outer state, $VASLA_i$ is dormant. In this case, FSLA is responsible for choosing the next action.

Algorithm 1 action_selection()

Notation: $FSLA$ denotes a fixed structure learning automaton with $L_{KN,K}$,
$VASLA_i$ 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 FSLA,
V_i denotes the last action made by the i^{th} VASLA

```
1:  Begin
2:    if FSLA.is_action_switching() = True: then
3:      if N_L=1: then
4:        V_L = VASLA_L.action_selection(['Grow', 'Stop'])
5:      else
6:        V_L = VASLA_L.action_selection(['Grow','Stop','Shrink'])
7:      end if
8:      switch (V_L)
9:      case 'Grow':
10:       N_L = N_L + 1
11:       FSLA.update_structure(L, N_L)
12:     case 'Shrink':
13:       N_L = N_L - 1
14:       FSLA.update_structure(L, N_L)
15:     case 'Stop':
16:       /*Do nothing about structure*/
17:     end switch
18:     L = FSLA.action_selection()
19:     return L
20:   else
21:     L = FSLA.action_selection()
22:     return L
23:   end if
24: End
```

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.
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.

Algorithm 2 update(β)

Notation: $FSLA$ denotes a fixed structure learning automaton with $L_{KN,K}$,
$VASLA_i$ 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 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
    FSLA.update(0)
  end if
End
```

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

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

- If 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 (FSLA may reaches the depth state or stay in it), depth_counter and transition_counter variables will increase by one.
  - Otherwise, the depth transition does not occur, therefore FSLA just receives the reward and goes to the inner state. transition_counter variable increases due to performing an action, and depth_counter variable does not alter. As a result, the probability of being in depth will decrease, and the conducted VASLA is prone to be penalized.
- Or else, AVDHLA is penalized ($\beta = 0$). This causes the FSLA to be penalized as well. In this situation, FSLA moves toward the outer state. transition_counter increases, and depth_counter stays the same as before. Consequently, the probability of being in 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 outer environment can be formulated as a triple $< \alpha, \beta, C >$ in which [10]:

1. $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ denotes a finite set of the inputs.
2. $\beta = \{\beta_1, \beta_2\}$ denotes a binary set of the outputs. For mathematical convenience, β_1 and β_2 are chosen to be 0 and 1 respectively.

3. $C = \{c_1, c_2, ..., c_r\}$ 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:

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

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

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

From an operational standpoint [10], 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 [10, 32].

6.1.2. Non-Stationary Environment

Contrary to the stationary environment, whenever c_i varies over time, the environment is non-stationary [10].

If a learning automaton with a fixed strategy is used in such an environment, it may becomes 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 leads 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 $E_i \in \{E_1, E_2, ..., E_d\}$. 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 [10, 32].

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 with the stage number t, such a composite environment is referred to as a *State-Dependent Environment* [10, 32].

In this environment, changes of the states may rely on the t^{th} stage explicitly or implicitly. Generally, this kind of environment is described by the c_i penalty probabilities.

State-dependent environments have three well-known models (A, B, C). In this study, 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 in the t^{th} stage, two events will occur:

1. The c_i probability of the t^{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 $\psi_j(t)$ $(i, j = 1, 2, ..., r)$ are both non-negative functions of the t^{th} stage. 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 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:

$$\psi_j(t) = \begin{cases} \psi_j & c_j(t) - \psi_j(t) \geq 0 \\ 1 - c_j(t) & o.w \end{cases} \tag{13}$$

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 [10, 32, 75]:

- **Total Number of Rewards (TNR)**: This evaluation metric shows the total number of rewards that the automaton has received from the environment ($\beta = 1$). In the i^{th} iteration, 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 is shown in the following equation [75]:

$$TNR_i = \sum_{j \leq i} \beta_j \tag{14}$$

- **Total Number of Action Switching (TNAS)**: This metric is used to count how many times the automaton should change the chosen action in an environment. This metric is so important because some learning automata cost a lot for changing the action. As a matter of fact, TNAS_i shows the aggregation of the action switching of the automaton at the i^{th} stage. The equation for TNAS_i is presented as follows [75]:

$$TNAS_i = \sum_{j \leq i} s_j \tag{15}$$

  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 [75].

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 toward 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 (Ex1)**: The main goal of this experiment is to evaluate the proposed automaton in an environment with fixed penalty probability (c_i).
2. **Experiment2 (Ex2)**: This category of experiments is dedicated to testing the proposed automaton in the Markovian switching environments with different properties for each environment.
3. **Experiment3 (Ex3)**: 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 stage.

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, rounded up to the nearest integer. 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 is for investigating the learning capability of the automaton.

- Experiment 1.2 is designed to compare the proposed automaton with FSLA.

- Experiment 1.3 studies the proposed automaton versus VSLA.

- Experiment 1.4 aims to compare our proposed automaton with HLA, a leading method in the $L_{KN,K}$ field.

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

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 in each instant or not.

To examine the learning capability of 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.

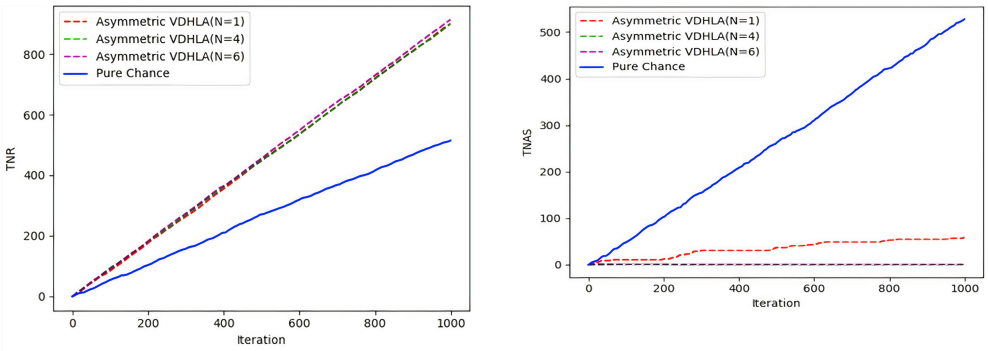
For the sake of explanation, the first action (action α_1) has a probability of 0.1. This means that the automaton in this environment will be rewarded in 10% of the time by choosing α_1. Meanwhile, choosing the second one (action α_2) brings the performed action of the automaton with a reward in 90% 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 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. Figure.7 shows the result from the TNR and TNAS aspects.

As these figures show, AVDHLA outperforms the pure chance automaton with respect to TNR and TNAS. This leads us to the result that AVDHLA is able to learn effectively. Furthermore, according to the results of this experiment, 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 ability of the automaton to increase its depth for the favorable action. Actually, the automaton with higher values of depth has a better initial condition, and therefore needs less effort to change the depth. Eventually, they will obtain more rewards and decrease their action switching number towards the favorable action.



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

Figure 7: Experimental results of Ex 1.1 with reward probability vector of (0.1, 0.9)

Two other configurations are considered in Appendix 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 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 (K), 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.

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

The considered initial depth of AVDHLA is 1, 3, 5, and 7 for configuration numbers 1 to 4, respectively. The number of iterations in the AVDHLA is 10000 because, 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 show that in most scenarios AVDHLA outperforms FSLA in both TNR and TNAS aspects. This indicates that AVDHLA is less sensitive than FSLA concerning the initial depth. This feature shows itself in lower values of depth

Table 2: Experimental Results of Experiment 1.2 for 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

since AVDHLA can increase its depth autonomously. If the depth of $L_{KN,K}$ increases, it can exploit more effectively towards 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 will engage in excessive exploration, leading to a decrease in the number of rewards obtained.

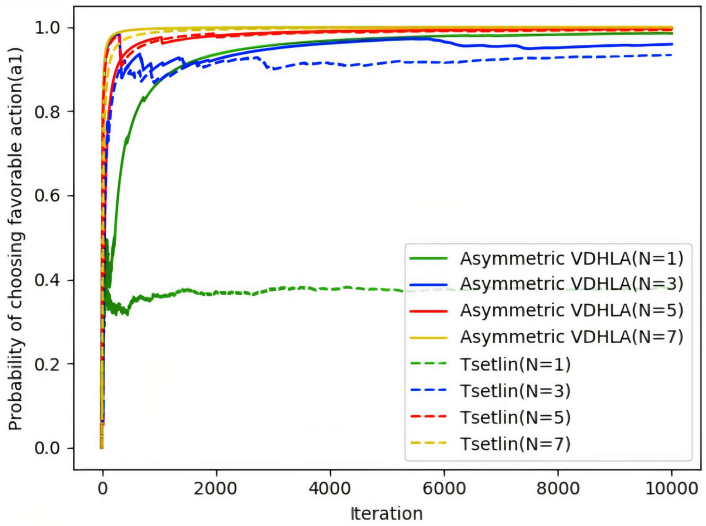


Figure 8: Experimental results of Ex 1.2 with respect to the probability of choosing the favorable action for AVDHLA with K = 9

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 remark that more experiments have been performed to have a better comparison between the AVDHLA and FSLA. These experiments exist in Appendix Appendix B.2.

Experiment 1.3. An experiment needs to be designed to compare the AVDHLA with 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. Equation 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 VSLAs to compare with VDHLA. Each configuration is tested with two values for λ_1 and λ_2 (denoted by "a" and "b"). These configurations are displayed in Table.3 from number 1 to 5.

Table 3: The Setup of Experiment 1.3

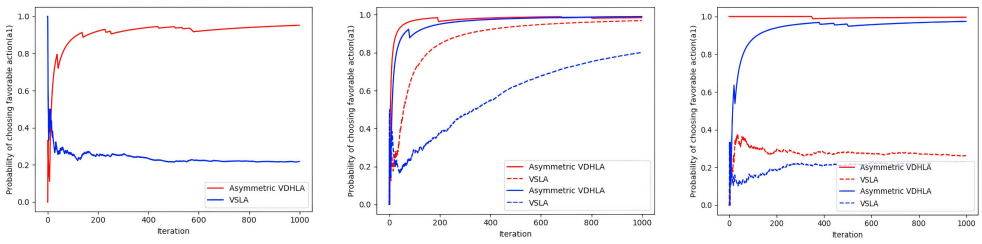
	P	L_{R-I}		L_{P-I}		L_{R-P}		$L_{R-\varepsilon P}$	
	Config1	Config2.a	Config2.b	Config3.a	Config3.b	Config4.a	Config4.b	Config5.a	Config5.b
K	5	5	5	5	5	5	5	5	5
N	5	5	5	5	5	5	5	5	5
λ_1	0	0.1	0.01	0	0	0.1	0.01	0.1	0.01
λ_2	0	0	0	0.1	0.01	0.1	0.01	0.01	0.001

The results of this experiment with respect to TNR and TNAS are shown in Table.4, considering 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 VSLA in all cases. It means that AVDHLA, unlike VSLA, doesn't have any dependencies on updating schemes. Also, the lower number of action switching indicates that the proposed automaton learns to find the favorable action.

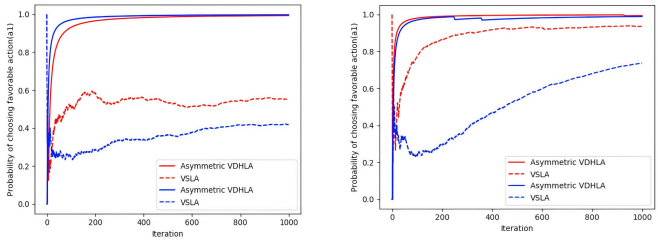
Table 4: Experimental results of Ex 1.3 with respect to TNR and TNAS

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	$\mathbf{1.60 * 10^{-132}}$	$\mathbf{1.09 * 10^{-140}}$	**0.005**	$\mathbf{1.70 * 10^{-12}}$	$\mathbf{1.41 * 10^{-130}}$	$\mathbf{5.10 * 10^{-153}}$	$\mathbf{3.38 * 10^{-129}}$	$\mathbf{2.39 * 10^{-156}}$	$\mathbf{1.74 * 10^{-128}}$	$\mathbf{4.21 * 10^{-154}}$	$\mathbf{5.63 * 10^{-78}}$	$\mathbf{9.70 * 10^{-103}}$	$\mathbf{1.15 * 10^{-92}}$	$\mathbf{2.79 * 10^{-118}}$	$\mathbf{1.55 * 10^{-9}}$	$\mathbf{2.26 * 10^{-21}}$	$\mathbf{5.26 * 10^{-81}}$	$\mathbf{2.48 * 10^{-81}}$

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



(a) Config 1 (b) Config 2 (c) Config 3



(d) Config 4 (e) Config 5

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

part of the experiment leads us to the result that the probability of choosing the favorable action in AVDHLA, in contrast to 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, AVDHLA demonstrates success. Moreover, the best performances are observed in L_{R-P} and $L_{R-\varepsilon P}$, 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 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, AVDHLA outperforms 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 HLA,

Table 5: Experimental Results of Experiment 1.4 for AVDHLA

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**

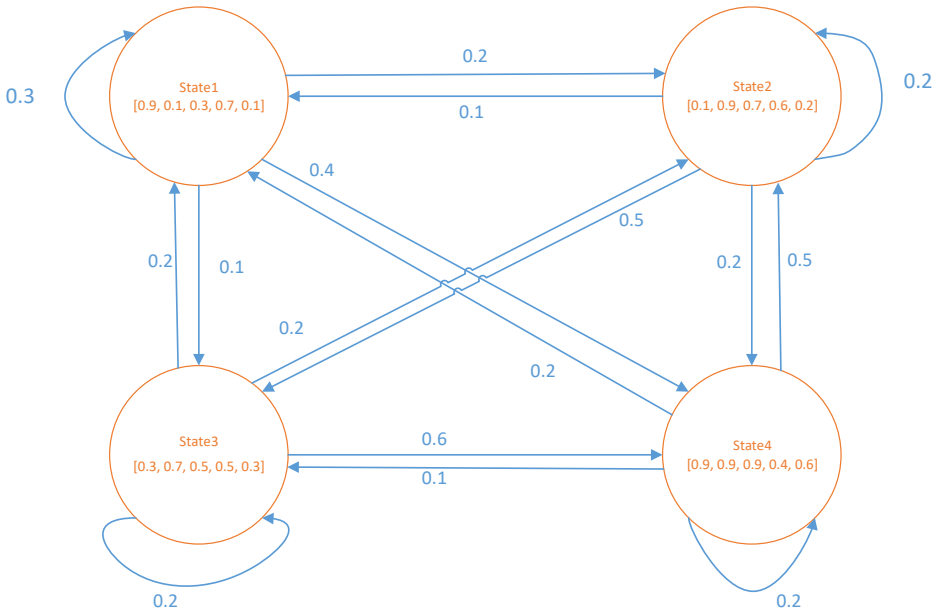


Figure 10: The designed environment for Markovian Switching experiment

AVDHLA dynamically sets its depth based on its interaction with the outer environment. This will help achieve a balance between exploration and exploitation. Additionally, in other scenarios, 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 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, the initial depth is 1, 3, 5, 7 for configurations 1 to 4 respectively, and 10000 iteration is considered for each experiment.

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

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

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

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

Experiment 2.1. This experiment is performed to compare AVDHLA with FSLA. The results are shown in Table.6. According to the fairly random designed environment, the proposed automaton can adapt itself to the Markovian switching environment with respect to TNR and TNAS. We should remark that all the actions have approximately the same probability of getting a reward in the designed environment, hence we should not except extremely high performance. In addition, AVDHLA outperforms FSLA in terms of obtaining rewards and minimizing action switching, except for Configuration 4. The primary reason for this lies in AVDHLA's ability to select the right depth through actions with a higher probability of reward. However, in the fourth configuration, AVDHLA explores more

to find an appropriate depth; consequently, its performance is lower than that of FSLA. Finally, we should conclude that AVDHLA can be trusted in Markovian switching environments.

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**

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

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	$\mathbf{6.34 * 10^{-8}}$

6.3.3. Experiment 3 - State-dependent Environment

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

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

For the above experiments, three tuples of (θ, ϕ) 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 FSLA in the State-dependent environment. The motivation is to analyze the effect of the ζ and ψ parameters introduced in the 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, 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.

Table 8: Experimental Results of the 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

In this experiment, action 1 is favored at the first stage. After 1000 iterations, this action remains favorable although ζ and ψ affect the reward probability vector. The results in Table.8 reveal the superiority of AVDHLA over FSLA with respect to TNR and TNAS in most configurations except configuration 3. In configuration 3, the initial depth is set at 5, seemingly an appropriate value for the automaton. However, AVDHLA further explores additional depths through action switching in pursuit of identifying the most suitable 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 AVDHLA. For VSLA, the experiment is set up exactly as in Equation 16. The inner VASLAs have the same configurations as VSLA (except for the number of actions), as shown in Table.9.

According to the results of this experiment, are shown in Table.10, AVDHLA outperforms VSLA in most configurations with respect to TNR and TNAS. When

Table 9: The Setup of VSLA in Experiment 3.2

	P	L_{R-I}		L_{P-I}		L_{R-P}		$L_{R-\varepsilon P}$	
	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

both λ_1 and λ_2 are equal to 0 and AVDHLA turns into FSLA, it doesn't change its decision easily, resulting in more rewards and fewer action switches. Additionally, among the various configurations, L_{R-P} is superior because it is sensitive to penalties. Therefore, receiving penalties in an environment will affect the probability vector effectively and cause AVDHLA to set the depth appropriately. Ultimately, AVDHLA is mostly superior to VSLA in State-dependent environments with respect to the defined experiment metrics.

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**	$\mathbf{1.23 * 10^{-5}}$	$\mathbf{6.65 * 10^{-21}}$	$\mathbf{2.31 * 10^{-72}}$	$\mathbf{2.46 * 10^{-135}}$	$\mathbf{2.18 * 10^{-115}}$	$\mathbf{1.54 * 10^{-107}}$	$\mathbf{2.60 * 10^{-120}}$	$\mathbf{1.36 * 10^{-61}}$	$\mathbf{9.02 * 10^{-90}}$	$\mathbf{3.87 * 10^{-71}}$	$\mathbf{4.57 * 10^{-99}}$	**0.004**	$\mathbf{6.73 * 10^{-17}}$	$\mathbf{6.28 * 10^{-47}}$	$\mathbf{1.63 * 10^{-76}}$

Experiment 3.3. This experiment is designed to compare AVDHLA with 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, AVDHLA outperforms HLA regarding TNR and TNAS. It is evident that whenever the depth of $L_{KN,K}$ is not appropriate, due to the changing attributes of the environment, AVDHLA will choose a depth in a way that maximizes its reward.

Table 11: Experimental Results of the 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**	$\mathbf{1.91 * 10^{-8}}$

7. Application : Defense Against Selfish Mining Attack

In this section, AVDHLA is used to design a novel defense mechanism, named Nik [46, 76, 77], against the selfish mining [78, 45, 79] attack in Bitcoin [44, 80, 81]. 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 automaton in such a complex environment are presented.

7.1. Selfish Mining Concepts

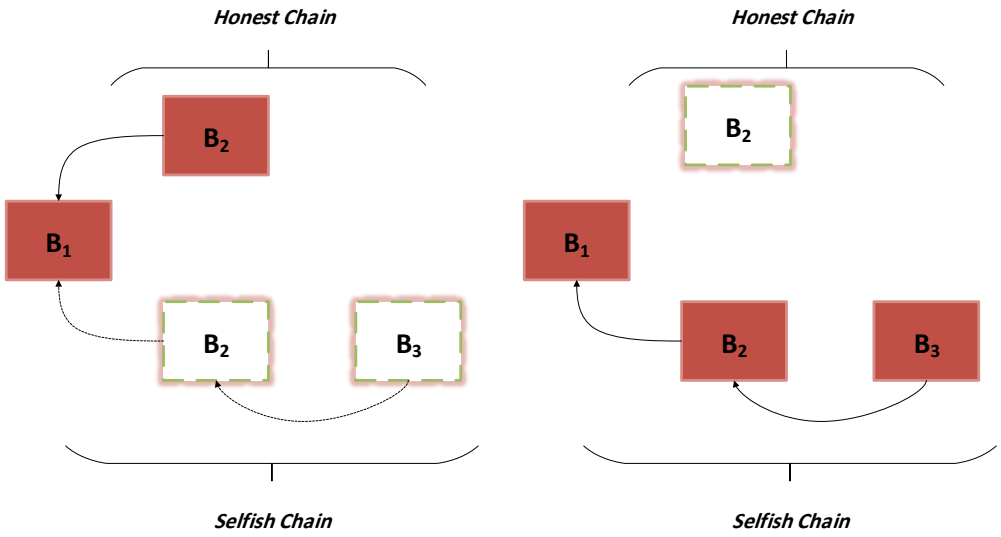
Bitcoin [44] is a decentralized cryptocurrency that was introduced by Satoshi Nakamoto in 2009. It has received a lot of attention due to its decentralized nature [44, 80].

The transactions in the Bitcoin network are recorded through units called blocks. Creating a new block requires effort to solve a cryptopuzzle with a dedicated reward. The participants who put their resources to solve such a puzzle are called miners [44, 80, 45, 46].

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 [80].

However, handling Bitcoin in a decentralized manner is a challenging problem. Some attacks like selfish mining [45] 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



(a) Stage 1 (b) Stage 2

Figure 11: A scenario of the selfish mining attack

right work, will be discarded. Consequently, a consensus is reached on the selfish branch of the fork [82, 83].

To enhance our comprehension of selfish mining, let's 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 white 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 white 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 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.

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

```
1: Begin
2:    if (ρ = ρ_max) then
3:       L = LA.choose _ action(['Stop', 'Shrink'])
4:    else if (ρ = ρ_min) then
5:       L = LA.choose _ action(['Grow', 'Stop'])
6:    else
7:       L = LA.choose _ action(['Grow', 'Stop', 'Shrink'])
8:    end if
9:    switch (L)
10:   case 'Grow':
11:      ρ = ρ + 1
12:   case 'Shrink':
13:      ρ = ρ − 1
14:   case 'Stop':
15:      /*Do nothing about ρ*/
16:   end switch
17: End
```

7.2. Proposed AVDHLA-Based Defense

In this section, we will describe our novel defense mechanism which takes advantage of 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.
- 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.

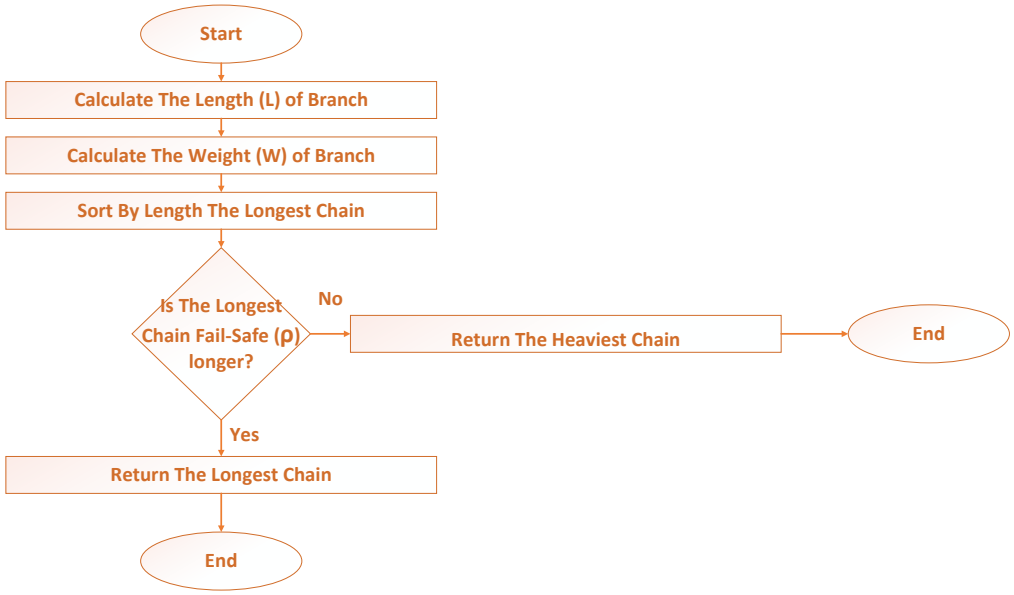


Figure 12: The flowchart of the algorithm to aid the miner in decision-making

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 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 automaton should receive feedback from the environment. We have designed a virtual environment for the 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 automaton.

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

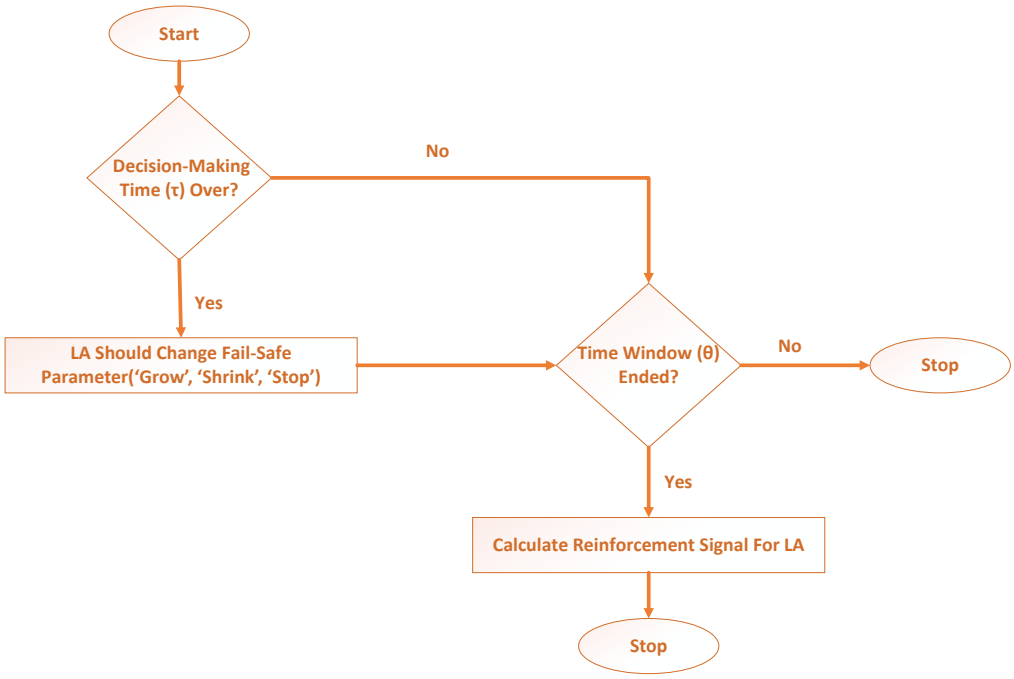


Figure 13: The flowchart of modifying fail-safe parameter and how the automaton receives feedback

7.3. Selfish Mining Metrics

Two metrics are considered to evaluate the performance of the 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:

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

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

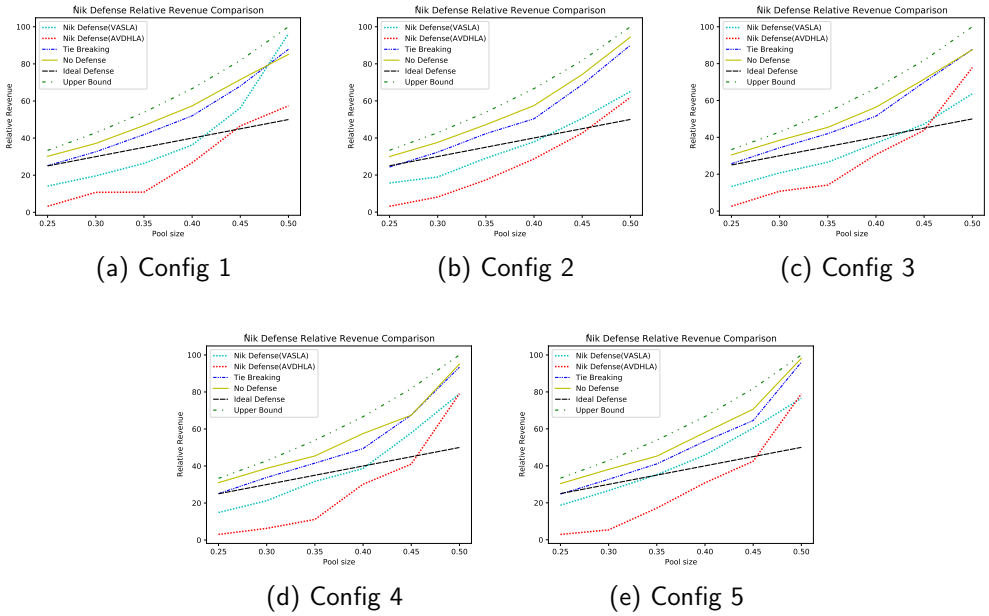
This experiment aims to provide a comparison of the proposed defense using a more sophisticated learning automaton with the well-known defense, namely tie-breaking. In tie-breaking, when a miner is aware of the fork, they choose one of the branches of that fork in a uniform random manner.

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.

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

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



(a) Config 1 (b) Config 2 (c) Config 3

(d) Config 4 (e) Config 5

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

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 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- $L_{R-\varepsilon P}$ 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, AVDHLA outperforms the others in conjunction with relative revenue. Both pieces of evidence lead us to the outstanding performance of the automaton in the complex environment like the blockchain. Additionally, it shows that 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 AVDHLA increases the lower bound threshold from 0.25 up to 0.4 approximately.

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

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

Eventually, we can trust 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 with respect to the introduced metrics in Appendix Appendix D.

8. Application : Recommendation Systems

In this section, 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 [84, 85], 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 [86, 87] such as streaming services [88], e-commerce [89], and social media [90]. 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 [91]: 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.

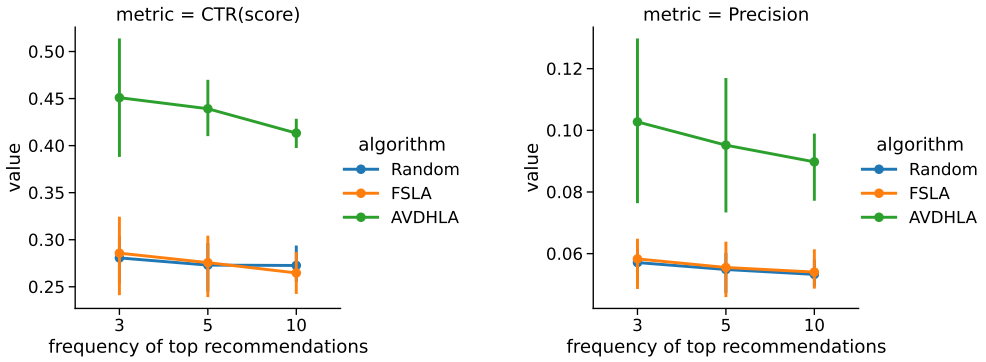
One of the most significant challenges faced by recommender systems is managing the balance between exploration and exploitation. Exploration involves recommending new or less-explored content to discover new user preferences, while exploitation focuses on delivering content that aligns with known user 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] [92, 93, 94] 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 [92, 93, 94], 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 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.

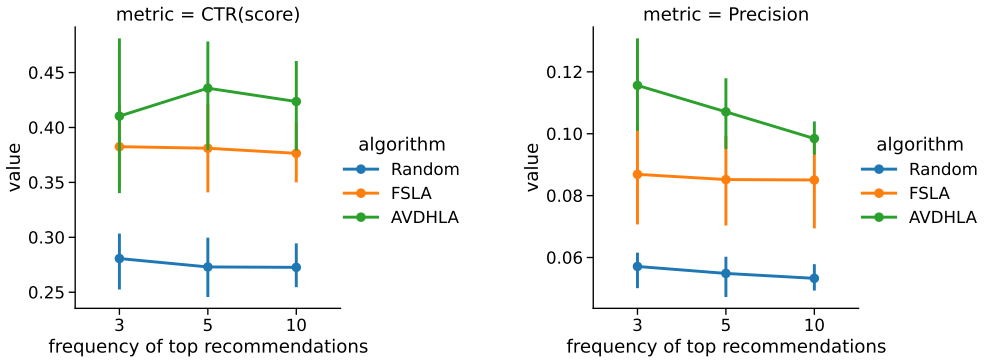
CTR [95, 96] is a key metric in evaluating the effectiveness of a recommendation system. It measures the proportion of times a user clicked on a recommended item, calculated as the number of clicks divided by the total number of recommendations presented to the user. This metric directly reflects how relevant the recommendations are to users, with a higher CTR indicating more successful en-

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



(a) CTR (b) Precision

Figure 15: Experimental results of recommendation systems for $N = 1$



(a) CTR (b) Precision

Figure 16: Experimental results of recommendation systems for $N = 3$

gagement.

Precision [95, 96] 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 AVDHLA and FSLA in the evaluation. The internal VASLAs in 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 Fig.15 and Fig.16, demonstrate that AVDHLA consistently outperforms FSLA in terms of both CTR and Precision. The superior performance of AVDHLA can be attributed to its ability to dynamically adjust its depth, optimizing the balance between exploration and exploitation. For $N = 1$, 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 $N = 3$, FSLA improves performance through increased exploitation. Nevertheless, AVDHLA shows a clear advantage as it avoids the limitations faced by FSLA by adjusting depth more intelligently to adapt to different scenarios.

It's worth noting that precision rates of 5% are generally considered high for most recommender systems. In this case, the learning automaton-based recommender, particularly 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

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

diverse scenarios. Additionally, we explore potential avenues for improvement. The ensuing points encapsulate our analysis:

- A key differentiator for the proposed method compared to its predecessors is its ability to dynamically adjust its depth through interactions with the surrounding environment. This adaptive feature sets our approach apart, enhancing its versatility and responsiveness.

- The implementation of AVDHLA is so straightforward. It can be reproduce just by combining $L_{KN,K}$ and some VASLAs so easily just as it illustrated in the 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 $L_{KN,K}$ due to the utilization of 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 $L_{KN,K}$, as each VASLA requires a 1-D vector with three inner items to store 'Grow,' 'Shrink,' and 'Stop' probabilities.

- Expanding the number of actions within our algorithm will inherently escalate its learning complexity, thereby leading to an increase in learning time complexity.

- Improperly configuring the reward and penalty rate parameters can significantly impact the performance of 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 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 AVDHLA be applied in secure data transmission and trustworthiness frameworks, such as the approach proposed by Jiang et al [97] against cyber-physical attacks. AVDHLA can serve as a decision-maker, effectively 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 [98, 99, 100, 101].

- As the prevailing solutions against selfish mining predominantly rely on reinforcement learning [82, 83], considering the contemporary trend, it is plausible to explore alternative AI methods like [102, 103] for addressing this issue.

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 $L_{KN,K}$. 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 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, 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 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, AVDHLA performed successfully compared to FSLA in the designed experiments.

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

We are hopeful that 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 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.

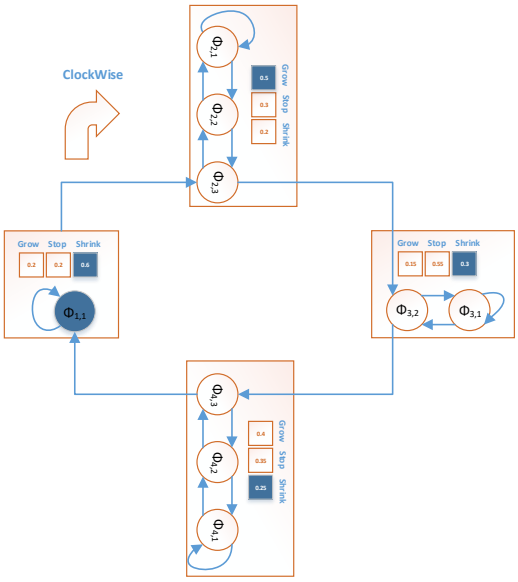
11. Funding

This research received no external funding.

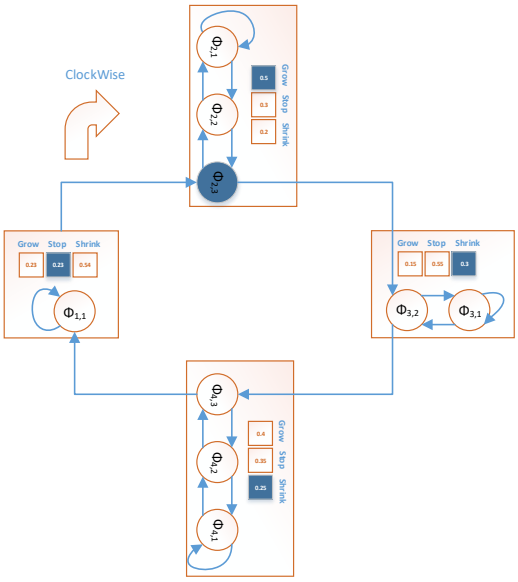
Appendix A. An Illustrative Example

To provide an example for the proposed 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.

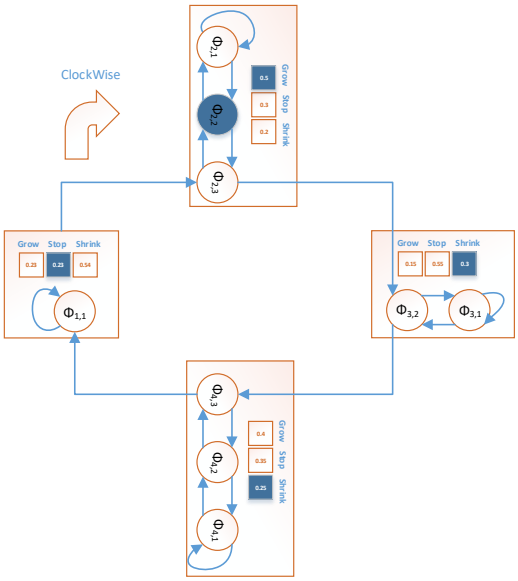
This scenario has four stages. The used AVDHLA is constructed with $AVDHLA$ $(K = 4, N_1 = 1, N_2 = 3, N_3 = 2, N_4 = 3, \lambda_1 = 0.5, \lambda_2 = 0.5)$. This automaton has 4 actions. Each action has its own depth (1 for action 1, 3 for action 2, 2 for action 3, and 3 for action 4). Also, each action has a specific VASLA with a particular



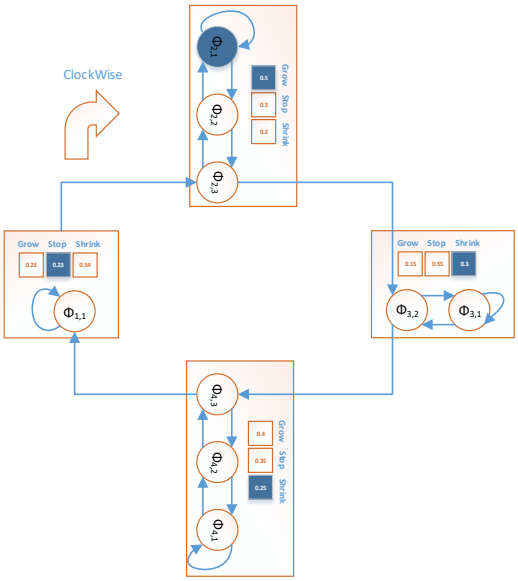
(a) Stage one of the example



(b) Stage two of the example



(c) Stage three of the example



(d) Stage four of the example

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

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 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, 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 VASLA_1 is 0. This shows that the last action was not good.

The probability vector of 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 Equation.1 for the 'Shrink' action, and the Equation.2 for the other actions. These changes are applied to Fig A.17.b, which is related to the current stage.

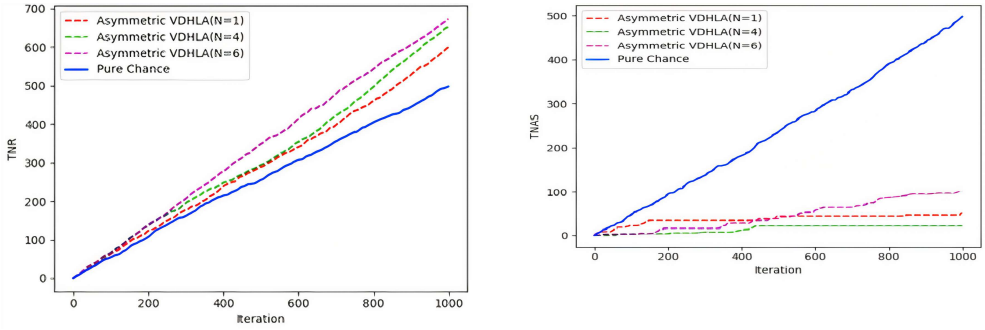
The VASLA_1 needs to omit the 'Shrink' action from its options for choosing the next action because the depth of action 1 is equal to 1, and if the 'Shrink' action is chosen, the next depth will be 0, leading to an invalid result. Under such circumstances, VASLA_1 undertakes to choose between the 'Grow' or 'Stop' options. We assume that the 'Stop' option is chosen by VASLA_1.

In the third stage, the 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.

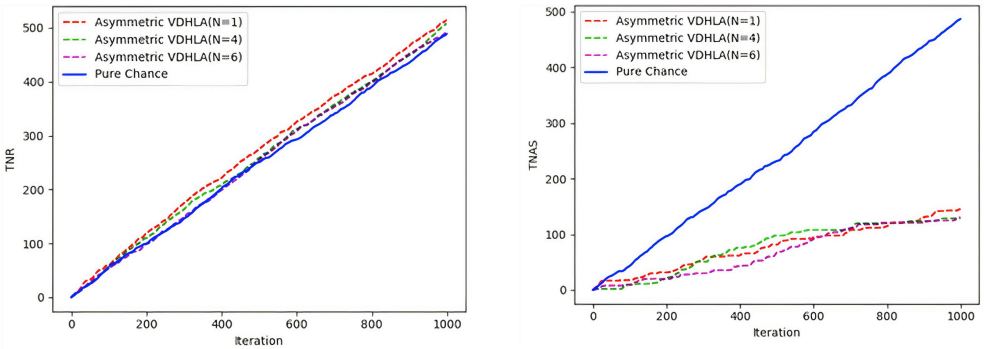
To have a better overview, it is repeated once again. The FSLA deserves the reward from the environment, remains in state $\phi_{(2,1)}$. Similar to stage 4, both the $transition-counter$ and $depth-transition$ will increase because there is a transition, and that transition is of the depth-transition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

This scenario will continue, and as it can be seen, the proposed automaton can synchronize itself with the outer environment to obtain the highest possible



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

Figure A.18: Experimental results of Ex 1.1 with reward probability vector of (0.3, 0.7)



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

Figure A.19: Experimental results of Ex 1.1 with reward probability vector of (0.5, 0.5)

number of rewards.

Appendix B. Exploring the Stationary Environment: Additional Experiment Insights

Appendix B.1. Experiment 1.1 Extras

Extra experiments were designed to learn more about the capabilities of the new 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 Fig.A.18 and Fig.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 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 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 automaton with $N = 1$ experiences depth changes aimed at increasing its depth.

In Fig.A.19(a), given equal probabilities for both actions, the automaton with the lowest depth exhibits superior adaptability to the environment, resulting in higher rewards. Additionally, Fig.A.19(b) 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 AVDHLA can adapt itself to the outer environment. Because AVDHLA increases its depth decisively, as a result, it does not change the chosen action easily.

Appendix B.2. Experiment 1.2 Extras

Completing our comparison of the AVDHLA family with 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 FSLA with respect to TNR and TNAS. Lower values of K can help the 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 automaton finds the favorable action among the lower number of actions more simply.

Table B.13: Experimental Results of Experiment 1.2 for AVDHLA

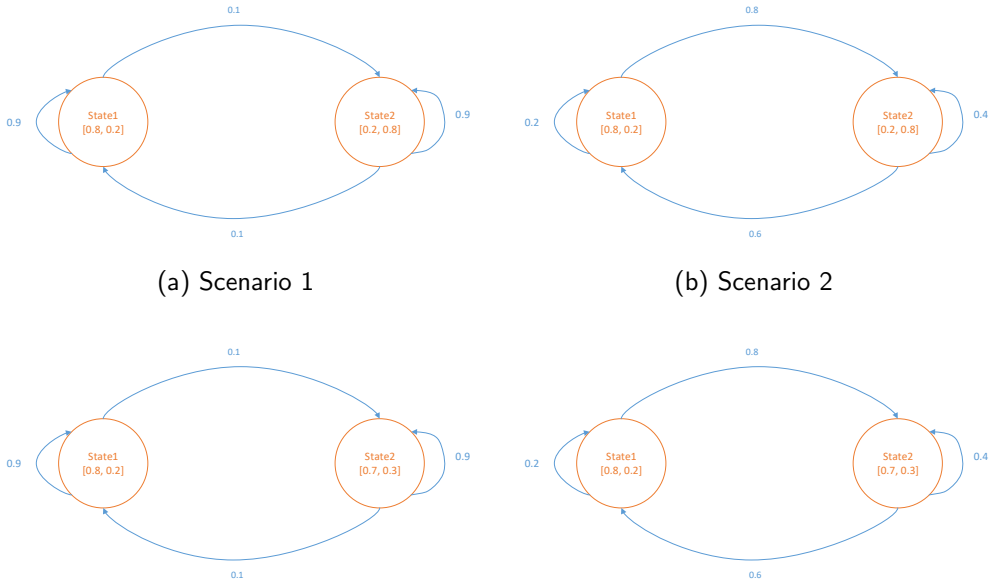
Model	Config 1		Config 2		Config 3		Config 4	
	TNR	TNAS	TNR	TNAS	TNR	TNAS	TNR	TNAS
				K = 2				
AVDHLA	**8032**	**31**	**8075**	**25**	**8045**	**8**	**8000**	**1**
FSLA	6736	3264	7986	142	7953	31	8000	4
				K = 5				
AVDHLA	**7934**	**123**	**8018**	**37**	**7990**	**26**	**8008**	**7**
FSLA	4600	5400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

Appendix C. Exploring the Markovian Switching Environment: Additional Experiment Insights

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

For a better understanding of the traits of 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



(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Figure C.20: Four scenarios of Markovian Switching environment

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 in the four scenarios have a common property such as: 2, 5, 9 are considered for the number of allowed actions (K), inner VASLAs are L_{R-I} with $\lambda_1 = 0.01$ and $\lambda_2 = 0$, and the initial depth is 1, 3, 5, 7 for configurations 1 to 4 respectively. All of the reported results are for 10000 iterations.

Appendix C.1. Scenario 1

The aim of this scenario is to evaluate the proposed 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.

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

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:

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

Multiplication of the steady state matrix to the reward probability matrix of Markovian switching environment yields to 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, an 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.

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

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, AVDHLA outperforms FSLA with respect to TNR and TNAS because of the flexibility of this automaton to choose the desired depth for each action separately. Overall, we cannot expect these automata to learn from this environment.

Appendix C.2. Scenario 2

This scenario is conducted to study the proposed 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:

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

Multiplication of the steady state matrix to the reward probability matrix of Markovian switching environment yields to the reward probability matrix of the equivalent stationary environment.

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

Therefore, the second action is more likely to be rewarded by the outer 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.

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

The results (Table.C.15) and mathematical analysis show that AVDHLA outperforms FSLA with respect to TNR and TNAS. This performance can be seen at lower initial depths. The underlying reason is the confusion of the 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 automaton comes at a lot of cost. Overall, we can infer that in such environments where the favorable action changes periodically, AVDHLA might be a good choice.

Appendix C.3. Scenario 3

The purpose of this scenario is to assess the proposed 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 stationary environment depends on multiplication of the steady state matrix to reward probability matrix of 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.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

The results (Table.C.16) show the superiority of AVDHLA over 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 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 AVDHLA can asymmetrically choose its depth relative to FSLA. As expected, this automaton works better in such an environment in conjunction with the experiment metrics.

Appendix C.4. Scenario 4

This scenario is conducted to evaluate the proposed 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.

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

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 the 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), AVDHLA outperforms FSLA with respect to TNR and TNAS. In such an environment with existence of the favorable action, AVDHLA can increase its depth through the favorable action. As a result,

it will collect more rewards with less number of action switching. Eventually, AVDHLA can be robust in this environment in conjunction with the considered metrics.

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

Appendix D. Nik Defense Experiment Extras

This appendix is dedicated to perform extra experiments on the proposed defense in the existence of AVDHLA as a decision maker. Three scenarios of are considered as follows:

1. This scenario examines the effect of 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, 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 $L_{R-\varepsilon P}$ with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$.

Appendix D.1. Scenario 1

This scenario is conducted to examine the proposed defense with various values of ρ to see the effect 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 to perform the experiment. ρ values can vary discretely between the minimum value and the max discreetly. The value of τ is equal to the time it takes to mine five blocks, and each θ is divided into ten time intervals.

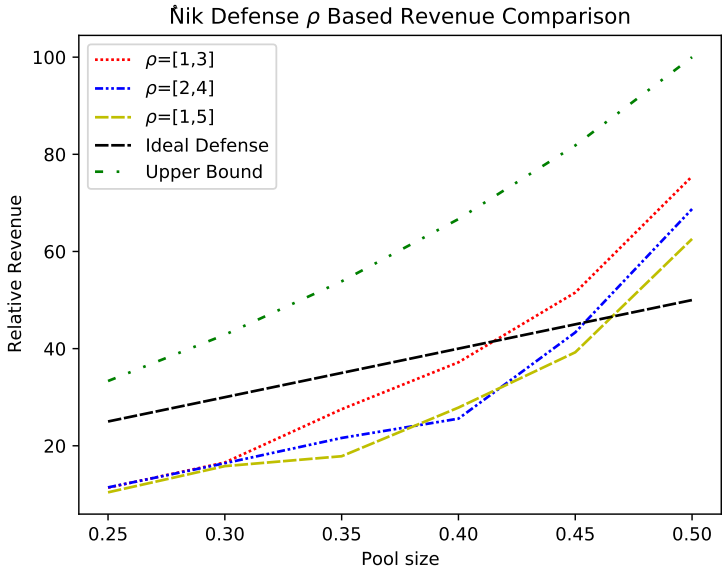


Figure D.21: Experimental results of Scenario 1 with respect to the relative revenue

The results in Fig.D.21 demonstrate the superiority of $[1,5]$ for AVDHLA with respect to the relative revenue. It shows the tendency of the automaton toward the higher values of ρ. In AVDHLA, the 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 show the same result for the lower bound threshold. Overall, we can claim that longer values of the fail-safe interval are better for AVDHLA because of the freedom to choose asymmetrically in relation to the experiment metrics.

Appendix 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.

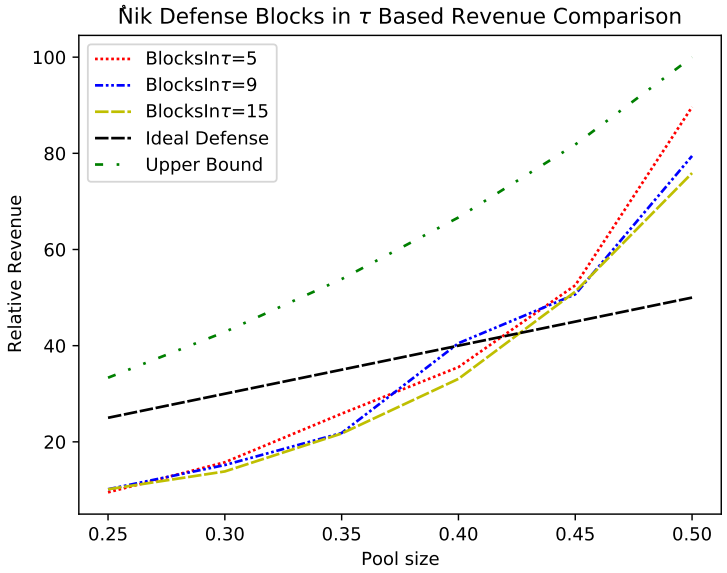


Figure D.22: Experimental results of Scenario 2 with respect to the relative revenue

The obtained results are displayed in Fig.D.22. From these results, one can observe that the lower values of τ lead to less relative revenue for the selfish miners. This is close to our expectations, since the number of τ in one θ is constant. On the other hand, lower values of τ lead to an increase in the number of θ in one simulation. It means that the automaton gets more feedback from the environment. As a result, it will tune the β parameter effectively.

Appendix 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.

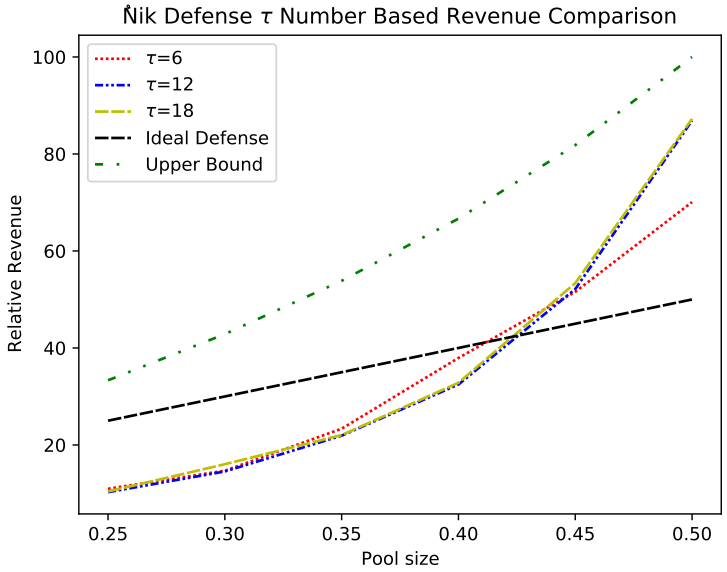


Figure D.23: Experimental results of Scenario 3 with respect to the relative revenue

By looking at the results in Fig.D.23 for 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 automaton, the less number of τ in one θ leads to more number of θ in one simulation. This means that the automaton can adapt itself to the 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.

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, Frontiers of Information Technology & Electronic Engineering 21 (12) (2020) 1726–1744.

[4] S. Gronauer, K. Diepold, Multi-agent deep reinforcement learning: a survey, Artificial Intelligence Review (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, Advances in Neural Information Processing Systems 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, arXiv preprint arXiv:2306.03346 (2023).

[9] M. A. Thathachar, P. S. Sastry, Varieties of learning automata: an overview, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 32 (6) (2002) 711–722.

[10] K. S. Narendra, M. A. Thathachar, Learning automata: an introduction, Courier corporation, 2012.

[11] L. Jiao, X. Zhang, O.-C. Granmo, K. D. Abeyrathna, On the convergence of tsetlin machines for the xor operator, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (5) (2022) 6072–6085.

[12] X. Zhang, L. Jiao, O.-C. Granmo, M. Goodwin, On the convergence of tsetlin machines for the identity-and not operators, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (10) (2021) 6345–6359.

[13] 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.

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

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

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

[17] 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.

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

[19] A. B. Hashemi, M. R. Meybodi, A note on the learning automata based algorithms for adaptive parameter selection in pso, Applied Soft Computing 11 (1) (2011) 689–705.

[20] K. S. Narendra, K. Parthasarathy, Learning automata approach to hierarchical multiobjective analysis, IEEE Transactions on systems, man, and cybernetics 21 (1) (1991) 263–272.

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

[22] A. Reina, Robot teams stay safe with blockchains, Nature Machine Intelligence 2 (5) (2020) 240–241.

[23] A. Yazidi, D. Silvestre, B. J. Oommen, Solving two-person zero-sum stochastic games with incomplete information using learning automata with artificial barriers, IEEE Transactions on Neural Networks and Learning Systems (2021).

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

[25] 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 Communications magazine 44 (12) (2006) 51–57.

[26] 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 Transactions on Neural Networks and Learning Systems 32 (8) (2020) 3444–3457.

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

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

[29] Z. Zhang, D. Wang, J. Gao, Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks, IEEE transactions on neural networks and learning systems 32 (10) (2020) 4639–4652.

[30] 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.

[31] 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.

[32] M. A. Thathachar, P. S. Sastry, Networks of learning automata: Techniques for online stochastic optimization, Springer Science & Business Media, 2003.

[33] M. Thathachar, P. S. Sastry, A new approach to the design of reinforcement schemes for learning automata, IEEE transactions on systems, man, and cybernetics (1) (1985) 168–175.

[34] G. I. Papadimitriou, M. Sklira, A. S. Pomportsis, A new class of/spl epsi/-optimal learning automata, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (1) (2004) 246–254.

[35] M. Thathachar, B. R. Harita, Learning automata with changing number of actions, IEEE transactions on systems, man, and cybernetics 17 (6) (1987) 1095–1100.

[36] O.-C. Granmo, The tsetlin machine–a game theoretic bandit driven approach to optimal pattern recognition with propositional logic, arXiv preprint arXiv:1804.01508 (2018).

[37] S. Leonardos, G. Piliouras, K. Spendlove, Exploration-exploitation in multi-agent competition: convergence with bounded rationality, Advances in Neural Information Processing Systems 34 (2021) 26318–26331.

[38] 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.

[39] Q. Chen, Q. Zhang, Y. Liu, Balancing exploration and exploitation in episodic reinforcement learning, Expert Systems with Applications 231 (2023) 120801.

[40] X. He, K. Zhao, X. Chu, Automl: A survey of the state-of-the-art, Knowledge-based systems 212 (2021) 106622.

[41] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, Advances in neural information processing systems 28 (2015).

[42] J. M. Kübler, V. Stimper, S. Buchholz, K. Muandet, B. Schölkopf, Automl two-sample test, Advances in Neural Information Processing Systems 35 (2022) 15929–15941.

[43] 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.

[44] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Decentralized business review (2008).

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

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

[47] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et al., Model-based reinforcement learning: A survey, Foundations and Trends® in Machine Learning 16 (1) (2023) 1–118.

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

[49] 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.

[50] J. Ramírez, W. Yu, A. Perrusquía, Model-free reinforcement learning from expert demonstrations: a survey, Artificial Intelligence Review (2022) 1–29.

[51] 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.

[52] A. Rezvanian, A. M. Saghiri, S. M. Vahidipour, M. Esnaashari, M. R. Meybodi, Recent advances in learning automata, Vol. 754, Springer, 2018.

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

[54] R. Seraj, J. Sharma, O.-C. Granmo, Tsetlin machine for solving contextual bandit problems, Advances in Neural Information Processing Systems 35 (2022) 30194–30205.

[55] 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.

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

[57] 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.

[58] 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.

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

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

[61] 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.

[62] K. Darshana Abeyrathna, O.-C. Granmo, X. Zhang, L. Jiao, M. Goodwin, The regression tsetlin machine: a novel approach to interpretable nonlinear regression, Philosophical Transactions of the Royal Society A 378 (2164) (2020) 20190165.

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

[64] 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.

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

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

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

[68] 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, The Computer Journal 67 (2) (2024) 508–518.

[69] S. Ghanavati, J. Abawajy, D. Izadi, Automata-based dynamic fault tolerant task scheduling approach in fog computing, IEEE Transactions on Emerging Topics in Computing 10 (1) (2020) 488–499.

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

[71] F. Safara, A. Souri, S. F. Deiman, Super peer selection strategy in peer-to-peer networks based on learning automata, International Journal of Communication Systems 33 (6) (2020) e4296.

[72] 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 Transactions on Neural Networks and Learning Systems 32 (8) (2020) 3444–3457.

[73] A. Yazidi, D. Silvestre, B. J. Oommen, Solving two-person zero-sum stochastic games with incomplete information using learning automata with artificial barriers, IEEE Transactions on Neural Networks and Learning Systems 34 (2) (2021) 650–661.

[74] C. Dey, R. Bose, K. K. Ghosh, S. Malakar, R. Sarkar, Lagoa: Learning automata based grasshopper optimization algorithm for feature selection in disease datasets, Journal of Ambient Intelligence and Humanized Computing (2022) 1–20.

[75] S. Gholami, A. M. Saghiri, S. Vahidipour, M. Meybodi, Hla: a novel hybrid model based on fixed structure and variable structure learning automata, Journal of Experimental & Theoretical Artificial Intelligence 35 (2) (2023) 231–256.

[76] 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. doi:10.5220/0012378600003636.

[77] 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, arXiv preprint arXiv:2302.12096 (2023).

[78] I. Eyal, The miner's dilemma, in: 2015 IEEE symposium on security and privacy, IEEE, 2015, pp. 89–103.

[79] 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.

[80] 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.

[81] 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.

[82] T. Wang, S. C. Liew, S. Zhang, When blockchain meets ai: Optimal mining strategy achieved by machine learning, International Journal of Intelligent Systems 36 (5) (2021) 2183–2207.

[83] 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.

[84] S. S. Khanal, P. Prasad, A. Alsadoon, A. Maag, A systematic review: machine learning based recommendation systems for e-learning, Education and Information Technologies 25 (4) (2020) 2635–2664.

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

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

[87] 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 Transactions on Services Computing 13 (4) (2020) 685–695.

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

[89] 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 and Applications 83 (16) (2024) 48521–48537.

[90] 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.

[91] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, S. Luo, A review of content-based and context-based recommendation systems, International Journal of Emerging Technologies in Learning (iJET) 16 (3) (2021) 274–306.

[92] 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.

[93] E. Strong, B. Kleynhans, S. Kadıoğlu, Mabwiser: parallelizable contextual multi-armed bandits, International Journal on Artificial Intelligence Tools 30 (04) (2021) 2150021.

[94] 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.

[95] 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.

[96] 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.

[97] 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 Transactions on Systems, Man, and Cybernetics: Systems 52 (12) (2022) 7799–7809.

[98] B. Eysenbach, T. Zhang, S. Levine, R. R. Salakhutdinov, Contrastive learning as goal-conditioned reinforcement learning, Advances in Neural Information Processing Systems 35 (2022) 35603–35620.

[99] 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, Computers in Biology and Medicine 150 (2022) 106054.

[100] J. Zhou, Z. Wu, Z. Jiang, K. Huang, K. Guo, S. Zhao, Background selection schema on deep learning-based classification of dermatological disease, Computers in Biology and Medicine 149 (2022) 105966.

[101] 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.

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

[103] Y. Wang, T. Bai, T. Li, L. Huang, Osteoporotic vertebral fracture classification in x-rays based on a multi-modal semantic consistency network, Journal of Bionic Engineering 19 (6) (2022) 1816–1829.