The Journal of Supercomputing
https://doi.org/10.1007/s11227-020-03293-z

Improving learning ability of learning automata using chaos theory

Bagher Zarei[1] · Mohammad Reza Meybodi[2]

Abstract
A learning automaton (LA) can be considered as an abstract system with a finite set of actions. LA operates by choosing an action from the set of its actions and applying it to the stochastic environment. The environment evaluates the chosen action, and automaton uses the response of the environment to update its decision-making method for selecting the next action. This process is repeated until the optimal action is found. The learning algorithm (learning scheme) determines how to use the environment response for updating the decision-making method to select the next action. In this paper, the chaos theory is incorporated with the LA and a new type of LA, namely chaotic LA (cLA), is introduced. In cLA, the chaotic numbers are used instead of the random numbers when choosing the action. The experiment results show that in most cases, the use of chaotic numbers leads to a significant improvement in the learning ability of the LA. Among the chaotic maps investigated in this paper, the Tent map has better performance than the other maps. The convergence rate/convergence time of the LA will increase/decrease by 91.4%/29.6% to 264.4%/69.1%, on average, by using the Tent map. Furthermore, the chaotic LA has more scalability than the standard LA, and its performance will not decrease significantly by increasing the problem size (number of actions).

Keywords Reinforcement learning · Learning automata · Chaos theory · Chaotic map · Chaotic learning automata

✉ Mohammad Reza Meybodi
mmeybodi@aut.ac.ir

[1] Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

[2] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

Published online: 22 April 2020

1 Introduction

The process of learning by living organisms is a new research topic. The studies on this topic are divided into two general categories. The first category identifies the learning principles and its stages while the second category tries to suggest a methodology by which the principles can be included in a machine. Learning is generally defined as "some changes made in the performance of a system based on past experiences". An important characteristic of the learning systems is the ability to improve their performance over time. Mathematically, we can say that the goal of a learning system is to optimize a task that is not known completely. An approach is to reduce the goals of a learning system to an optimization problem, which is defined on a set of parameters, and its goal is to find the optimal parameter sets [1]. Since, in most problems, we have no information about the correct answers, a learning approach called reinforcement learning (RL) has been taken into consideration. RL is not a subset of neural networks, nor is it an alternative to them. Rather, it is an orthogonal approach to solve different and more difficult problems. RL combines the fields of dynamic programming and supervised learning to yield robust machine-learning systems. In RL, the learning agent is simply given a goal to achieve. The learning agent then learns how to achieve that goal by trial-and-error interactions with its environment [2]. The efficiency of these interactions will be evaluated by a numerical penalty that is taken from the environment [3].

The main advantage of RL compared to other learning approaches is that it requires no information about the environment except for the reinforcement signal [4]. One of the RL's methods is stochastic learning automata (SLA). Stochastic automaton tries to find the optimal action without any information about it (i.e., by considering the same probability for all the actions at the beginning). An action will be chosen randomly and applied to the environment. Then, the environment response will be taken, and the probability of the actions will be updated based on the learning algorithm, and this process will be repeated. Stochastic automaton which operates as the above to increase its performance is called SLA [4].

One of the main difficulties in using LA in many practical applications is the slow convergence speed and low accuracy. Although several attempts are conducted to increase the convergence speed and accuracy, these are not enough [4]. It has been shown that using chaotic numbers instead of random numbers leads to a significant improvement in the convergence and performance of an evolutionary algorithm [5, 6]. In [6], we proposed a chaotic memetic algorithm, namely CMA-net, that uses chaotic numbers at different stages of the evolutionary process. The results of experiments and statistical tests have shown that the CMA-net is more efficient than the standard memetic algorithm (memetic algorithm without using chaotic numbers) and other evolutionary algorithms. This motivated us to incorporate chaos theory with the LA and evaluate the impact of using chaotic numbers on the learning ability of the LA. The main contributions of this paper are summarized as follows:

- To the best of our knowledge, incorporation of chaos theory with the LA is proposed for the first time in this paper. The *ergodicity* property of chaos theory causes the LA performance to be significantly improved. In this paper, the effect of fourteen one-dimensional chaotic maps (see Table 1) on the convergence rate, convergence speed, and accuracy of the LA is investigated.

- Simulation results on five typical problems, corresponding to five LAs with 5, 10, 20, 50, and 100 actions demonstrate the superiority of the proposed chaotic LA in terms of convergence rate, convergence speed, and accuracy compared with the standard LA. Furthermore, the chaotic LA has more scalability than the standard LA, and its performance will not decrease significantly by increasing the problem size (number of actions).

The rest of this paper is organized as follows: The learning automata theory, chaos theory, and problem statement are given in Sect. 2. In Sect. 3, some of the related works are briefly reviewed. The proposed chaotic learning automata and experimental results are presented in Sects. 4 and 5, respectively. Finally, Sect. 6 contains the conclusion of the paper.

2 Preliminaries

2.1 Learning automata theory

Reinforcement learning (RL) or learning with a critic is defined by characterizing a learning problem, instead of characterizing the learning methods [3]. Any method that is well suited to solve that problem is considered as a RL method. The RL problem is the problem of learning from interactions in order to achieve a specific goal. Considering this specification for the RL problem, there must be an agent capable of learning, called the learner or the decision-maker. The learner must interact with a so-called environment or teacher, comprising everything outside of the learner, which provides evaluative responses to the actions performed by the learner. The learner and the environment interact continually; the learner selects actions and the environment responds to the selected actions, presenting new situations to the learner. In short, RL is a framework of the learning problems in which the environment does not indicate the correct action, but provides only a scalar evaluative response to the selection of an action by the learner.

Learning automaton (LA) as one of the computational intelligence techniques has been found a handy tool to solve many complex and real-world problems in networks where a large amount of uncertainty or lacking the information about the environment exists [7, 8]. A LA is a stochastic model operating in the framework of the RL [9]. This model can be classified under the RL schemes in the category of the temporal-difference (TD) learning methods. TD learning is a combination of the Monte Carlo ideas and dynamic programming ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like the dynamic programming, TD methods update estimates based in part on the other learned estimates, without waiting for a final outcome [3]. Sarsa [10], Q-learning [11], Actor-Critic methods [12], and R-learning [13] are other samples of TD methods. Las differ from other TD methods in the following two ways: the representation of the internal states and the updating method of the internal states.

The automata approach to learning can be considered as the determination of an optimal action from a set of actions. A learning automaton can be regarded as an abstract object that has a finite number of actions. It selects an action from its finite set of actions and applies it to an environment. The environment evaluates the applied action and sends a reinforcement signal to the learning automaton (Fig. 1). The reinforcement signal provided by the environment is used to update the internal state of the learning automaton. By continuing this process, the learning automaton gradually learns to select the optimal action, which leads to favorable responses from the environment.

A learning automaton is a quintuple $\{\alpha, \Phi, \beta, F, G\}$ where $\alpha = \left\{\alpha_1, \alpha_2, \ldots, \alpha_r\right\}$ is the set of actions that it must choose from, $\Phi = \left\{\Phi_1, \Phi_2, \ldots, \Phi_s\right\}$ is the set of states, $\beta = \left\{\beta_1, \beta_2, \ldots, \beta_m\right\}$ is the set of inputs, $F : \Phi \times \beta \rightarrow \Phi$ is the state transition function that defines the transition of the state of the learning automaton upon receiving an input from the environment, and $G : \Phi \rightarrow \alpha$ is the output function and determines the action taken by the learning automaton.

The selected action at the instant n, denoted by $\alpha(n)$, serves as the input to the environment, which in turn emits a stochastic response, $\beta(n)$, at the instant n, which is considered as the response of the environment to the learning automaton. Based on the nature of β, environments could be classified into three classes: P-, Q-, and S-models. The output of a P-model environment has two elements of success or failure. Usually, in P-model environments, a failure (or unfavorable response) is denoted by 1, while a success (or a favorable response) is denoted by 0. In Q-model environments, β can take a finite number of values in the interval [0, 1] while in S-model environments, β lies in the interval [0, 1]. Based on the response $\beta(n)$, the state of the learning automaton $\Phi(n)$ is updated, and a new action is chosen at the instant $(n+1)$. Learning automata can be classified into two main classes: fixed structure learning automata (FSLA) and variable structure learning automata (VSLA) [4].

The learning automaton is called fixed-structure if the following are fixed: (1) the probability of the transition from one state to another state (the state transition function F) and (2) the action probability of any action in any state (the output function G). Examples of the fixed-structure LA are $L_{2N,2}$, $G_{2N,2}$, Krylov, and Krinsky [4], which have been used in many applications such as the adaptation of the back-propagation algorithm parameters.

A VSLA is represented by a 6-tuple $\{\alpha, \Phi, \beta, P, G, T\}$, where α is the action sets as output, Φ is a set of internal states, β denotes the set of inputs, P is the state probability vector governing the choice of state at each instant n, G is the

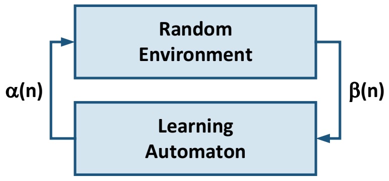


Fig. 1 Feedback connection of automaton and environment

output function, and T is the learning algorithm (also known as learning scheme). The learning algorithm T refers to a recurrence relation that is used to update the action probability vector. Let $\alpha_i(n) \in \alpha$ be the action that is chosen by a learning automaton, and $p(n)$ is the probability vector defined over the set of action at instant n. At each instant n, the action probability vector $p(n)$ is updated by the linear learning algorithm given in Eq. (1) if the chosen action $\alpha_i(n)$ is rewarded by the random environment ($\beta = 0$), and it is updated according to Eq. (2) if the chosen action is penalized ($\beta = 1$).

$$p_j(n+1) = \begin{cases} p_j(n) + a\left(1 - p_j(n)\right) & \text{if } j = i \\ (1-a)p_j(n) & \text{if } j \neq i \end{cases} \tag{1}$$

$$p_j(n+1) = \begin{cases} (1-b)p_j(n) & \text{if } j = i \\ \frac{b}{r-1} + (1-b)p_j(n) & \text{if } j \neq i \end{cases} \tag{2}$$

where a denotes reward parameter which determines the amount of increases of the action probability values, b is the penalty parameter determining the amount of decrease of the action probabilities values, and r is the number of actions that learning automaton can take. If $a = b$, the learning algorithm is called linear reward-penalty ($L_{R\text{-}P}$) algorithm; if $0 < b \ll a < 1$, the learning algorithm is called linear reward-ε-penalty ($L_{R\text{-}\varepsilon P}$) algorithm; and finally if $b = 0$, the learning algorithm is called linear reward-inaction ($L_{R\text{-}I}$) algorithm which is perhaps the earliest scheme considered in mathematical psychology.

2.2 Chaos theory

Chaos theory is a branch of mathematics that studies dynamical systems that follow deterministic laws but appear random and unpredictable. The hallmark of chaotic systems is their sensitivity to initial conditions. A small change in the initial conditions of such systems results in a massive change in long-term behavior. However, chaotic systems are deterministic, not random. Given the same initial conditions, the long-term results are repeatable. A system must have the following properties to be considered as a chaotic system [14–16].

Sensitivity to initial conditions: This property indicates that a slight change in the initial conditions of a chaotic system produces quite different results in the long-term. Therefore, a long-term prediction of the behavior of chaotic systems cannot be made in detail. Long-term predictions can be made to some extent, as long as vast spatial scales are involved. Although a chaotic system is defined by a deterministic equation, the sensitivity to the initial conditions is the only reason for this unpredictability.

Topological mixing or topological transitivity (ergodicity): This property indicates that the chaotic variables traverse all states non-repeatedly within a specific range according to its laws. Therefore, this property can be used as an optimization mechanism that ensures that no solution is visited in the search space twice and prevents from falling algorithm in the local optimum.

Topological density (density of periodic orbits): This property expresses that every point in the space is approached arbitrarily by periodic orbits.

In mathematics, chaotic maps are used to generate chaotic time series. One of the most commonly used chaotic maps is the Logistic map. The Logistic map is a second-order polynomial map with the following equation [17]:

$$X_{n+1} = rX_n\left(1 - X_n\right) \tag{3}$$

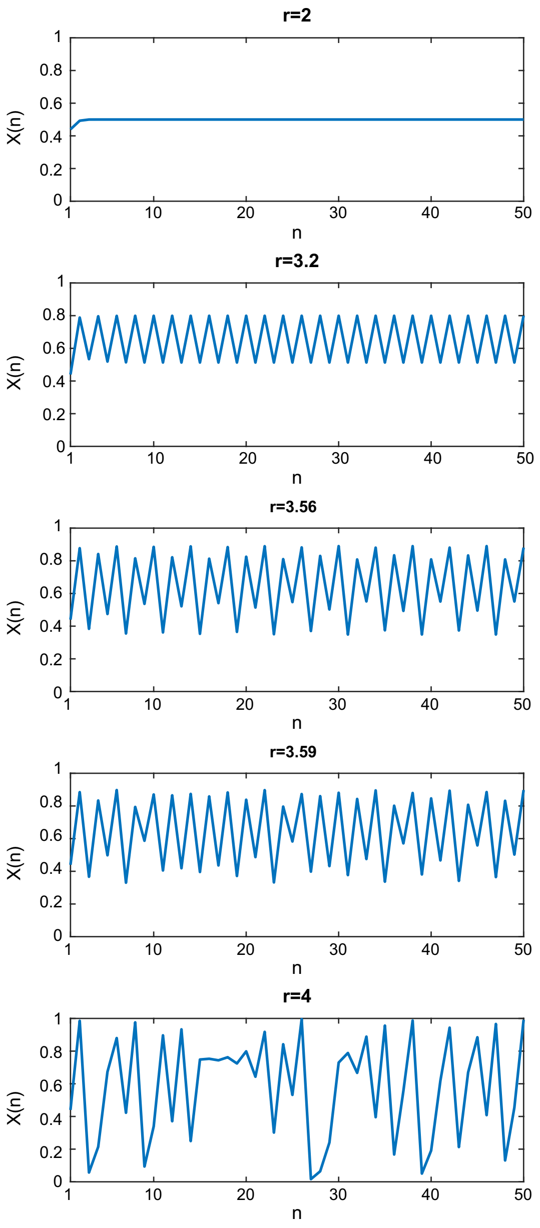
obviously, $X \in [0, 1]$ under the condition that initial $X_0 \in [0, 1]$ and that $X_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$. In the above equation, r is a control parameter in the range [0, 4] that determines the behavior of the variable X. The variable X can be *convergent*, *periodic*, or *chaotic*. When $0 \le r \le 3$, it converges to a constant number, when $3 < r \le 3.56$, it will show periodic behavior, and when $3.56 < r \le 4$, it will show chaotic behavior. In Fig. 2, the behavior of the variable X for different values of parameter r is shown.

A numerical approach to identify chaos can be developed based on the quantitative characterization of the divergence among the neighboring trajectories. If two trajectories that start close to each other deviate more and more with increasing time, the system is said to be chaotic. The rate at which nearby trajectories deviate from each other with time is characterized by a quantity called *Lyapunov exponent*. Negative Lyapunov exponents indicate convergence, while positive Lyapunov exponents demonstrate divergence and chaos. One can use the largest positive Lyapunov exponent (LLE) to detect the presence of chaos in a dynamical system [18]. In the Logistic map, the largest positive Lyapunov exponent is obtained when r is 4 [18]. In Table 1, the most common one-dimensional chaotic maps are listed.

2.3 Problem statement

As mentioned, a LA is defined as an automaton that improves its performance while operating in a random environment. The convergence speed and the average penalty are two significant quantities associated with the performance of an automaton [4]. The convergence speed is a measure of the *speed* of the automaton. It gives an indication of the time taken by the automaton to reach the stationary state. The average penalty, on the other hand, could be regarded as a measure of the *accuracy* of the automaton, for it gives an idea of the probability with which the desired action is selected. The closer average penalty is to the minimum penalty probability c_l, the more accurate is the automaton. c_l is the penalty probability of the optimal action by the environment. In practice, both these quantities are important. One would like to have maximum accuracy with the best possible speed. However, as is well known in systems theory, the two requirements frequently conflict, and it is necessary to compromise. For example, if the reward parameter a in Eq. (1) is too small, the learning process will be too slow. On the contrary, if the reward parameter a is too large, the increments in the action probabilities will be too high, and the automaton's accuracy in perceiving the optimal behavior becomes low. By choosing the parameter a to

Fig. 2 The behavior of the variable *X* in the Logistic map



be sufficiently small, the probability of convergence to the optimal behavior may be made as close to 1 as desired. Therefore, "*the problem is to propose a learning algorithm or any mechanism by which the LA can achieve the highest accuracy and maximum convergence speed.*"

Table 1 One-dimensional chaotic maps

No.	Name	Equation	Parameter(s)	Range
1.	Chebyshev [19]	$x_{n+1} = \cos\left(n\cos^{-1}\left(x_n\right)\right)$	None	$(-1, 1)$
2.	Bernoulli Shift [20]	$x_{n+1} = \begin{cases} \frac{x_n}{p} & 0 \le x_n \le p \\ \frac{x_n - p}{1-p} & p < x_n \le 1 \end{cases}$	$p = 0.9$	$(0, 1)$
3.	Cubic [21]	$x_{n+1} = px_n\left(1 - x_n^2\right)$	$p = 2.1$	$(0, 1)$
4.	ICMIC [20]	$x_{n+1} = \sin\left(\frac{p}{x_n}\right)$	$p = 0.2$	$(-1, 1)$
5.	Logistic [17]	$x_{n+1} = px_n\left(1 - x_n\right)$	$p = 3.9$	$(0, 1)$
6.	Piecewise [19]	$x_{n+1} = \begin{cases} \frac{x_n}{p} & 0 \le x_n < p \\ \frac{x_n - p}{0.5-p} & p \le x_n < 0.5 \\ \frac{1-p-x_n}{0.5-p} & 0.5 \le x_n < 1-p \\ \frac{1-x_n}{p} & 1-p \le x_n \le 1 \end{cases}$	$p = 0.4$	$(0, 1)$
7.	Sine [19]	$x_{n+1} = \frac{p}{4}\sin\left(\pi x_n\right)$	$p = 3.6$	$(0, 1)$
8.	Singer [19]	$x_{n+1} = p\left(7.86x_n - 23.31x_n^2 + 28.75x_n^3 - 13.302875x_n^4\right)$	$p = 1.05$	$(0, 1)$
9.	Sinusoidal [19]	$x_{n+1} = px_n^2\sin\left(\pi x_n\right)$	$p = 2.3$	$(0, 1)$
10.	Tent [21]	$x_{n+1} = \begin{cases} px_n & x_n < 0.5 \\ p\left(1 - x_n\right) & x_n \ge 0.5 \end{cases}$	$p = 1.9$	$(0, 1)$
11.	Circle [19]	$x_{n+1} = \left\{x_n + p_2 - \frac{p_1}{2\pi}\sin\left(2\pi x_n\right)\right\} \bmod 1$	$p_1 = 1$	$(0, 1)$
12.	Gaussian [18]	$x_{n+1} = \exp\left(-p_1 x_n^2\right) + p_2$	$p_1 = 7$	$(-1, 2)$
13.	Intermittency [19]	$x_{n+1} = \begin{cases} \varepsilon + x_n + cx_n^{p_2} & 0 < x_n \le p_1 \\ \frac{x_n - p_1}{1-p_1} & p_1 < x_n < 1 \end{cases}$ $c = \frac{\left(1-\varepsilon-p_1\right)}{p_1{}^{p_2}}$	$p_1 = 0.8$ $p_2 = 2$	$(0, 1)$
14.	Leibovitch [19]	$x_{n+1} = \begin{cases} ax_n & 0 < x_n \le p_1 \\ \frac{p_2 - x_n}{p_2 - p_1} & p_1 < x_n \le p_2 \\ 1 - b\left(1 - x_k\right) & p_2 < x_n < 1 \end{cases}$ $a = \frac{p_2}{p_1}\left(1 - \left(p_2 - p_1\right)\right)$ $b = \frac{1}{p_2 - 1}\left(p_2 - 1 - p_1\left(p_2 - p_1\right)\right)$	$p_1 = 0.6$	$(0, 1)$

3 Related works

Learning algorithms (reinforcement schemes) are generally classified into three categories:*non-estimator, estimator*, and *pursuit* [4, 8]. The non-estimator learning algorithms are, in turn, divided into three categories: *linear*, *nonlinear*, and *hybrid* [4, 8] (see Fig. 3). If $p(n + 1)$ is a linear function of $p(n)$, the reinforcement scheme is said to be linear; otherwise it is termed nonlinear. Sometimes, two or more schemes are combined to form a hybrid scheme. The aim of such a scheme is to realize the advantages of the constituent schemes, for example, speed of convergence or variance. In the following, each category is briefly described and some of proposed algorithms in each category are listed.

(A) Non-estimator learning algorithms: In non-estimator learning algorithms, the current value of the reinforcement signal is the only parameter that is used to update the action probability vector. Let us assume a finite action-set learning automaton (FALA) with r actions operating in a stationary P-model environment. α_i (for $i = 1, 2, \ldots, r$) denotes the action taken by this automaton at instant n (i.e. $\alpha(n) = \alpha_i$), and $\beta = \{0, 1\}$ is the response of the environment to this action. The following is a general learning algorithm for updating the action probability vector can be represented as follows:

$$p_j(n+1) = \begin{cases} p_j(n) - g_j\left[p(n)\right] & \text{when } \beta(n) = 0 \\ p_j(n) + h_j\left[p(n)\right] & \text{when } \beta(n) = 1 \end{cases} \tag{4}$$

for all $j \neq i$. For preserving probability measure, we must have $\sum_{j=1}^{r} p_j(n) = 1$, so we obtain

$$p_i(n+1) = \begin{cases} p_i(n) + \sum\limits_{\substack{j=1 \\ j\neq i}}^{r} g_j\left[p(n)\right] & \text{when } \beta(n) = 0 \\ p_i(n) - \sum\limits_{\substack{j=1 \\ j\neq i}}^{r} h_j\left[p(n)\right] & \text{when } \beta(n) = 1 \end{cases} \tag{5}$$

functions g_j and h_j (for $j = 1, 2, \ldots, r$) are called as reward and penalty functions, respectively. Reward and penalty functions are arbitrary, continuous, and nonnegative with the following properties.

$$\begin{aligned} &0 < g_j\left[p(n)\right] < p_j(n) \\ &0 < \sum_{\substack{j=1 \\ j\neq i}}^{r} \left[p_j(n) + h_j\left[p(n)\right]\right] < 1 \end{aligned} \tag{6}$$

for all $i = 1, 2, \ldots, r$ and all p whose elements are all in the open interval (0, 1). The continuity assumption on g_j and h_j is of mathematical convenience. The fact that both g_j and h_j are nonnegative maintains the reward and penalty nature of the updating. The property is shown in Eq. (6) ensures that all the components of $p(n+1)$ remain in (0, 1) when those of $p(n)$ are in the same open

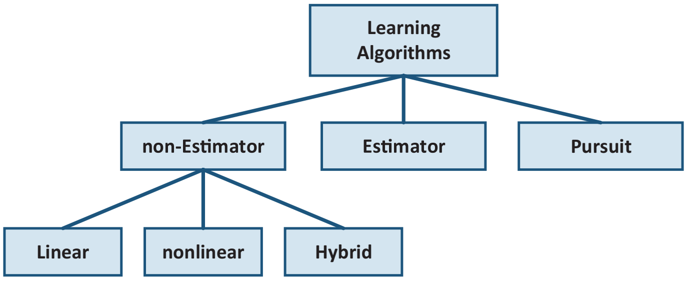


Fig. 3 Classification of learning algorithms

interval. Strict inequality is imposed in Eq. (6) so that $p(n+1) \neq p(n)$ when all the components of $p(n)$ are in the open interval (0, 1).

I. *Linear Learning algorithms*: In linear algorithms, the function under which the components of the action probability vector are updated is linear. Considerable attention has been paid to linear schemes on account of their analytical tractability. In these algorithms, the reward and penalty functions are defined as follows,

$$g_j\left[p(n)\right] = ap_j(n) \tag{7}$$

$$h_j\left[p(n)\right] = \frac{b}{r-1} + bp_j(n) \tag{8}$$

for all n and $j = 1, 2, \ldots, r$, where a and $b(0 \leq a, b < 1)$ represent the *reward* and *penalty* parameters, respectively. Different values of reward and penalty parameters result in different learning algorithms with different features. Five such linear schemes are the linear reward-penalty scheme ($L_{R\text{-}P}$), the linear reward-inaction scheme ($L_{R\text{-}I}$), the linear inaction-penalty scheme ($L_{I\text{-}P}$), the reward-ε-penalty ($L_{R\text{-}\varepsilon P}$), and the linear reward–reward scheme ($L_{R\text{-}R}$) [4, 8]. These schemes exhibit exciting and significantly different characteristics and, at present, are prototypes for distinct types of behavior observed in all learning automata [4].

II. *Nonlinear Learning algorithms*: Nonlinear algorithms include a wide variety of learning algorithms in which the rule under which the action probabilities are updated is nonlinear. Many studies have been conducted on linear algorithms, specifically $L_{R\text{-}P}$ and $L_{R\text{-}I}$ algorithms, for reasons of analytical simplicity, but nonlinear algorithms have not received the attention they merit. Two actions nonlinear learning algorithms are paid more attention, as it was not clear how the nonlinear increments were to be distributed among the various actions not selected by the automaton. Due to the nonlinearity of the updating rules, nonlinear learning algorithms cannot be easily generalized to more actions (e.g., three actions or more). Some of the nonlinear schemes are Vorontsova [22], Chandrasekharan and Shen [23], Shapiro and Narendra [24], Viswanathan and Narendra [25], Narendra and Thathachar [26], Meybodi and Lakshmivarahan [27], Thathachar and Oommen [28], and Poznyak and Najim [29].

III. *Hybrid Algorithms*: A hybrid learning algorithm is a combination of linear and nonlinear learning algorithms in which some advantages can be achieved due to the group synergic of algorithms. A well-defined combination of linear and nonlinear algorithms is useful to improve the convergence speed of algorithms in many cases. The property of the ε-optimality of a linear algorithm results in a low-speed convergence when the action probabilities approach their final values. On the other hand, the convergence speed of the expedient nonlinear algorithms is significantly faster as compared with ε-optimal algorithms. Hence, a combination of an ε-optimal algorithm and an expedient algorithm significantly improves the convergence speed to the optimal solution. Although many of these schemes exhibit a superior speed of operation and hence are of interest in practical application, the main problem with them is that it is difficult to get analytical results concerning aspects of their convergence. Viswanathan and Narendra [25] and Friedman and Shenker [30] are two hybrid schemes.

(B) Estimator learning Algorithms: One of the main difficulties in using LA in many practical applications is the slow rate of convergence and low accuracy. Estimator learning algorithms [31] are a class of the learning schemes which improve the convergence speed of the LA. In such algorithms, additional information about the characteristics of the environment is maintained and is used to increase the speed of convergence of the LA. Unlike other learning algorithms, in which the action probability vector is updated based solely on the current response received from the environment, in estimator learning algorithms, an estimation of the reward strength is maintained for each action of the LA and is used along with the current response of the environment to update the action probability vector. An estimator learning algorithm operates as follows. The learning automaton initially chooses an action (e.g., α_i). The selected action is applied to the environment. The random environment generates response $\beta(n)$ after it evaluates the chosen action. Learning automaton updates the action probability vector by using $\hat{d}(n) = \left[\hat{d}_1(n), \hat{d}_2(n), \ldots, \hat{d}_r(n)\right]$ and $\beta(n) \cdot \hat{d}(n)$ represents the set of reward estimations and $\hat{d}_j(n)$ denotes the estimation of the reward probability α_j at instant n. The estimation of the reward strength for any action α_j is computed as the ratio of the number of times α_j is rewarded to the number of times α_j is selected. $\hat{d}(n)$ is finally updated on the basis of the current response by the following rules.

$$\begin{aligned} R_i(n+1) &= R_i(n) + [1 - \beta(n)] \\ R_j(n+1) &= R_j(n), j \neq i \end{aligned} \tag{9}$$

$$\begin{aligned} Z_i(n+1) &= Z_i(n) + 1 \\ Z_j(n+1) &= Z_j(n), j \neq i \end{aligned} \tag{10}$$

$$\hat{d}(n) = \frac{R_k(n)}{Z_k(n)}, \quad k = 1, 2, \ldots, r \tag{11}$$

where $R_i(n)$ denotes the number of times action α_i is rewarded and $Z_i(n)$ denotes the number of times action α_i is selected. Different estimator learning algorithms have been proposed in the literature thus far, such as SELA [32], ASELA [33], DTSE [34], RRSA [35], and SEDEL [36].

(C) Pursuit learning Algorithms: A pursuit learning algorithm is a simplified version of the estimator learning algorithms, in which the action probability vector chases after the currently optimal action. At each iteration of a pursuit learning algorithm, the action probability of the action with the maximum rewarding estimation is increased. By this updating method, the learning algorithm always pursues the optimal action. Several different pursuit algorithms have been proposed in literature, namely pursuit reward-penalty ($P_{R\text{-}P}$) [37], discretized pursuit reward-penalty ($DP_{R\text{-}P}$) [38], pursuit reward-inaction ($P_{R\text{-}I}$) [38], discretized pursuit reward-inaction ($DP_{R\text{-}I}$) [39], generalized pursuit (GP) [40], and discretized pursuit generalized (DGP) [40].

4 Chaotic learning automata (cLA)

As mentioned, one of the main difficulties in using LA in many practical applications is the slow convergence speed and low accuracy. In the literature, several learning algorithms have been proposed to improve the convergence speed and accuracy of the LA. The idea of this paper is not to propose a new learning algorithm, but rather to use chaotic time series to increase the convergence speed and accuracy of the LA with existing learning algorithms. It has been shown that using chaotic numbers instead of random numbers leads to a significant improvement in the convergence and performance of an evolutionary algorithm [5, 6]. In [6], we proposed a chaotic memetic algorithm, namely CMA-net, that uses chaotic numbers at different stages of the evolutionary process. The results of experiments and statistical tests have shown that the CMA-net is more efficient than the standard memetic algorithm (memetic algorithm without using chaotic numbers) and other evolutionary algorithms. This motivated us to incorporate chaos theory with the LA and evaluate the impact of using chaotic numbers on the learning ability of the LA. The *ergodicity* property of chaos theory causes the convergence speed and accuracy of the LA to be significantly improved.

In the learning process of LA, random numbers are only used to select actions. The learning algorithm is definitive, and there are no random elements in it. The random numbers with uniform distribution in the range [0, 1] are used for selecting the actions. So, the main idea of this paper is to use chaotic numbers instead of random numbers for selecting the actions. We have called this type of LA chaotic LA (cLA). The difference between chaotic LA and standard LA is only in selecting the actions. In the standard LA, the random numbers and in the chaotic LA, the chaotic numbers are used for selecting the actions. It is expected that the learning ability of the LA improves by using chaotic numbers. This is due to the properties of the chaotic time series, especially the ergodicity property. The ergodicity property indicates that the chaotic variables traverse all states non-repeatedly within a specific range according to its laws.

For more explanation, consider a LA with four actions $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$. At first, the action probability vector of LA is $P = \{0.25, 0.25, 0.25, 0.25\}$. In the following, three cycles of the learning process of the chaotic LA have been explained. The Tent map with seed $x_0 = 0.31$ is used for generating the chaotic time series. The recursive relation of the Tent map is given in Table 1. The parameter p of this map is set to 1.9. Furthermore, the linear learning algorithm (Eqs. 1 and 2) is used to update the action probability vector. The reward and penalty parameters of the learning algorithm is set to 0.1.

Cycle 1 (instant $n = 1$)

(a) Selecting an action and applying it to the environment: The chaotic number $x_1 = 0.5270$ is generated by using the Tent map. According to the action probability vector of the LA, this number is equal to choose the action α_3 (probability proportionate selection or roulette wheel selection). The selected action will be applied to the environment. In Fig. 4, the way of generating the chaotic number x_1 and mapping it to the action α_3 is illustrated.

(b) Evaluating of the chosen action by the environment: The environment evaluates the action α_3 and the reinforcement signal β is given to the LA. Suppose $\beta = 0$ (reward).

(c) Updating the action probability vector: Since the environment has given a reward to the action α_3, the probability of this action will increase, and the probability of the other actions will decrease by using Eq. 1. The action probability vector will be $P = \{0.225, 0.225, 0.325, 0.225\}$ after updating the probability of the actions. In the below, the calculation of each element of this vector is given.

$$p_3 = p_3 + a\left(1 - p_3\right) = 0.25 + 0.1(1 - 0.25) = 0.325$$

$$p_1 = (1 - a)p_1 = (1 - 0.1)0.25 = 0.225$$

$$p_2 = (1 - a)p_2 = (1 - 0.1)0.25 = 0.225$$

$$p_4 = (1 - a)p_4 = (1 - 0.1)0.25 = 0.225$$

Cycle 2 (instant *n* = 2)

(a) Selecting an action and applying it to the environment: The chaotic number $x_2 = 0.8041$ is generated by using the Tent map. According to the action probability vector of the LA, this number is equal to choose the action α_4. The selected action will be applied to the environment. In Fig. 5, the way of generating the chaotic number x_2 and mapping it to the action α_4 is illustrated.

(b) Evaluating of the chosen action by the environment: The environment evaluates the action α_4 and the reinforcement signal β is given to the LA. Suppose $\beta = 1$ (penalty).

(c) Updating the action probability vector: Since the environment has given a reward to the action α_4, the probability of this action will increase, and the probability of the other actions will decrease by using Eq. 2. The action probability vector will be $P = \{0.2358, 0.2358, 0.3258, 0.2025\}$ after updating the probability of the actions. In the below, the calculation of each element of this vector is given.

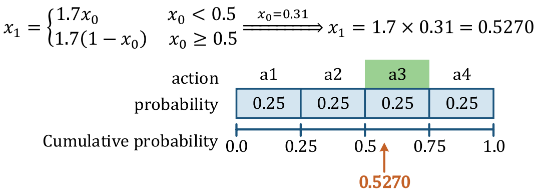

$x_1 = \begin{cases} 1.7x_0 & x_0 < 0.5 \\ 1.7(1 - x_0) & x_0 \geq 0.5 \end{cases} \xRightarrow{x_0=0.31} x_1 = 1.7 \times 0.31 = 0.5270$
action
a1
a2
a3
a4
probability
0.25
0.25
0.25
0.25
Cumulative probability
0.0
0.25
0.5
0.75
1.0
0.5270

Fig. 4 The way of generating the chaotic number x_1 and mapping it to the action α_3

$$p_4 = (1-b)p_4 = (1-0.1)0.225 = 0.2025$$

$$p_1 = \frac{b}{r-1} + (1-b)p_1 = \frac{0.1}{4-1} + (1-0.1)0.225 = 0.2358$$

$$p_2 = \frac{b}{r-1} + (1-b)p_2 = \frac{0.1}{4-1} + (1-0.1)0.225 = 0.2358$$

$$p_3 = \frac{b}{r-1} + (1-b)p_3 = \frac{0.1}{4-1} + (1-0.1)0.325 = 0.3258$$

Cycle 3 (instant *n* = 3)

(a) Selecting an action and applying it to the environment: The chaotic number $x_3 = 0.3330$ is generated by using the Tent map. According to the action probability vector of the LA, this number is equal to choose the action α_2. The selected action will be applied to the environment. In Fig. 6, the way of generating the chaotic number x_3 and mapping it to the action α_2 is illustrated.

(b) Evaluating of the chosen action by the environment: The environment evaluates the action α_2 and the reinforcement signal β is given to the LA. Suppose $\beta = 1$ (penalty).

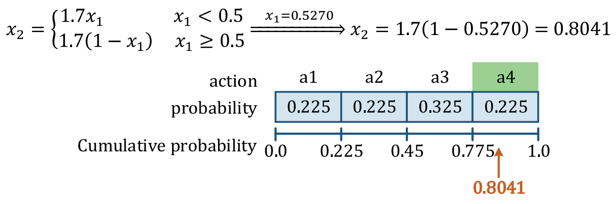
(c) Updating the action probability vector: Since the environment has given a reward to the action α_2, the probability of this action will increase, and the probability of the other actions will decrease by using Eq. 2. The action probability vector will be $P = \{0.2456, 0.2122, 0.3266, 0.2156\}$ after updating the probability of the actions. In the below, the calculation of each element of this vector is given.

$$p_2 = (1-b)p_2 = (1-0.1)0.2358 = 0.2122$$

$$p_1 = \frac{b}{r-1} + (1-b)p_1 = \frac{0.1}{4-1} + (1-0.1)0.2358 = 0.2456$$

$$p_3 = \frac{b}{r-1} + (1-b)p_3 = \frac{0.1}{4-1} + (1-0.1)0.3258 = 0.3266$$

$$p_4 = \frac{b}{r-1} + (1-b)p_4 = \frac{0.1}{4-1} + (1-0.1)0.2025 = 0.2156$$



$x_2 = \begin{cases} 1.7x_1 & x_1 < 0.5 \\ 1.7(1-x_1) & x_1 \geq 0.5 \end{cases} \xrightarrow{x_1=0.5270} x_2 = 1.7(1-0.5270) = 0.8041$
action
a1
a2
a3
a4
probability
0.225
0.225
0.325
0.225
Cumulative probability
0.0
0.225
0.45
0.775
1.0
0.8041

Fig. 5 The way of generating the chaotic number x_2 and mapping it to the action α_4

5 Experimental results and discussion

5.1 Experiments design

To evaluate the performance of the chaotic LA, five typical problems P_{I}, P_{II}, P_{III}, P_{IV}, and P_{V}, corresponding to five LAs with 5, 10, 20, 50, and 100 actions, respectively, have been selected for presenting the results of the simulation studies. In all problems, the linear learning algorithm presented in Eqs. 1 and 2 have been used for updating the action probability vector. The reward parameter a and the penalty parameter b are set to 0.1. Furthermore, the penalty probability of the optimal action by the environment is set to 0.1, and the penalty probability of the other actions (non-optimal actions) by the environment is set to 0.9. In all problems, the maximum number of trials is set to 50 times the number of actions of that problem. For example, in problem P_{I} we will have:

- $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$ is the set of LA actions: At instant t, the LA chooses one of these actions according to its action probability vector.
- $P = \{p_1, p_2, p_3, p_4, p_5\}$ is the LA action probability vector. p_i indicates the probability of choosing action a_i. The initial value of this vector is $P = \{0.2, 0.2, 0.2, 0.2, 0.2\}$. This vector is updated in the learning process according to the learning algorithm.
- In the learning algorithm, reward and penalty parameters are set to 0.1 ($a = b = 0.1$).
- $C = \{c_1, c_2, c_3, c_4, c_5\}$ is the set of environment penalty probabilities. c_i represents the probability that the application of action α_i to the environment will result in a penalty output. In the experiments, one random member of this set (corresponding to the optimal action) is set to 0.1, and the others (corresponding to the non-optimal actions) is set to 0.9. For example, the set can be $C = \{0.9, 0.1, 0.9, 0.9, 0.9\}$. In this case, action α_2 is the optimal action, and the probability of penalizing it by the environment is 0.1. Other actions (α_1, α_3, α_4 and α_5) are the non-optimal actions, and the probability of penalizing them by the environment is 0.9.
- The process of choosing an action, applying it to the environment, and updating the action probability vector is performed $5 \times 50 = 250$ times.

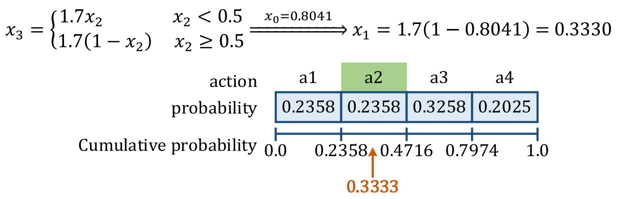

$x_3 = \begin{cases} 1.7x_2 & x_2 < 0.5 \\ 1.7(1-x_2) & x_2 \geq 0.5 \end{cases} \xrightarrow{x_0=0.8041} x_1 = 1.7(1-0.8041) = 0.3330$
action
a1
a2
a3
a4
probability
0.2358
0.2358
0.3258
0.2025
Cumulative probability
0.0
0.2358
0.4716
0.7974
1.0
0.3333

Fig. 6 The way of generating the chaotic number x_3 and mapping it to the action α_2

5.2 Evaluation criteria

The applicability of an automaton in a practical situation depends to a large extent on its speed of convergence as well as how closely it approaches optimal behavior. In fact, *speed* and *accuracy* generally serve as a basis of comparison of various schemes [4]. The convergence speed is a measure of the *speed* of the automaton. It gives an indication of the time taken by the automaton to reach the stationary state. The average penalty, on the other hand, could be regarded as a measure of the *accuracy* of the automaton, for it gives an idea of the probability with which the desired action is selected. The closer average penalty is to the minimum penalty probability c_l, the more accurate is the automaton. c_l is the penalty probability of the optimal action by the environment. In this paper, two criteria *convergence rate* (*CR*) [41] and *convergence time* (*CT*) [41] are used to evaluate the performance of the proposed chaotic LA. The *CR* is defined as the ratio between the number of experiments that LA converged correctly and the total number of experiments, while the *CT* as the average number of trials for a correct convergence. In the experiments, $p_{\text{opt}} = 0.95$ is considered as a condition of convergence. In other words, whenever the probability of the optimal action reaches 0.95, we say that the LA has been converged correctly. A LA with high CR and low CT has better learning ability.

$$\text{CR} = \frac{\text{the number of experiments that LA converged correctly}}{\text{total number of experiments}} \approx \text{accuracy} \tag{12}$$

$$\text{CT} = \text{the average number of trials for a correct convergence} \approx \frac{1}{\text{speed}} \tag{13}$$

5.3 Evaluation of the chaotic learning automaton (cLA)

In this section, the experimental results of the chaotic LA have been given. The process of learning in each problem has been repeated 1000 times (# of experiments = 1000)for different modes of selecting an action, and the results have been presented in Tables 2 and 3 and Figs. 7, 8, 9, 10, 11 and 12.

In Table 2, the convergence rate in the problems P_{I} to P_{V} with different chaotic maps are presented. The red cells indicate the chaotic LAs, which have worse performance than the standard LA. As you can see, the Chebyshev and Logistic maps in all the problems, the ICMIC map in the problems P_{II} to P_{V}, the Piecewise map in the problem P_{V} and the Sinusoidal map in the problems P_{I}, P_{II}, and P_{V} have worse performance than the standard LA. In other cases, the chaotic maps have better performance than the standard LA, and the convergence rate is more than the standard LA. For each problem, the best three chaotic maps are shown by a green background. As you can see, the Bernoulli Shift, Tent, and Leibovitch maps rank best in the problems II to V and the Tent, Circle and Gaussian maps in the problem I. According to the results, the convergence rate will increase by 61.2–264.4%, on average, when using the Bernoulli Shift, Tent and Leibovitch maps.

Table 2 Convergence rate in the problems P_I to P_V with different chaotic maps

	Action selection mode	Problem I (5 actions)	Problem II (10 actions)	Problem III (20 actions)	Problem IV (50 actions)	Problem V (100 actions)
Standard LA	Random	42.8	32.4	24.8	20.5	21.7
Chaotic LA	Chebyshev	2.7	1.1	0.5	0.3	0.5
	Bernoulli Shift	69.0	64.7	63.2	59.1	60.3
	Cubic	62.1	41.0	30.0	25.6	22.6
	ICMIC	51.8	29.4	19.5	13.6	8.4
	Logistic	22.1	18.2	15.1	11.7	11.1
	Piecewise	43.5	33.0	26.9	23.1	18.2
	Sine	64.8	57.6	45.0	34.6	30.9
	Singer	58.4	46.6	39.5	40.5	36.6
	Sinusoidal	41.0	30.6	24.8	22.1	20.1
	Tent	81.9	83.9	78.7	74.7	74.1
	Circle	80.2	62.1	50.9	44.4	42.7
	Gaussian	77.3	37.0	40.6	33.2	29.6
	Intermittency	53.3	46.7	40.1	40.5	38.2
	Leibovitch	72.3	66.0	62.3	62.1	63.8

Table 3 Convergence time in the problems P_I to P_V with different chaotic maps

	Action selection mode	Problem I (5 actions)	Problem II (10 actions)	Problem III (20 actions)	Problem IV (50 actions)	Problem V (100 actions)
Standard LA	Random	144.71	288.93	529.88	1307.98	2649.42
Chaotic LA	Chebyshev	140.93	307.18	490.60	1402.00	2743.20
	bernoulli shift	122.25	234.45	417.64	841.57	1501.92
	Cubic	63.23	74.61	96.54	170.39	393.84
	ICMIC	63.62	57.77	62.89	69.10	75.07
	Logistic	122.10	214.79	319.01	633.43	948.03
	Piecewise	143.78	279.93	507.47	1171.85	2400.39
	Sine	75.14	121.98	180.91	274.18	461.21
	Singer	81.14	106.29	152.51	297.89	501.78
	Sinusoidal	63.94	83.97	131.34	243.96	385.78
	Tent	101.89	160.93	234.88	445.35	819.88
	Circle	91.03	131.16	163.83	273.80	512.19
	Gaussian	84.54	62.30	99.82	115.00	127.46
	Intermittency	100.36	176.55	286.50	612.70	1037.34
	Leibovitch	111.55	203.00	289.72	518.17	887.39

In Table 3, the convergence time in the problems P_I to P_V with different chaotic maps are presented. The red cells indicate the chaotic LAs, which have worse performance than the standard LA. As you can see, all the chaotic maps have better performance than the standard LA in all the problems except the Chebyshev map. For each problem, the best three chaotic maps are shown by a green background. As you can see, the Cubic, ICMIC and Gaussian maps rank best in the problems II to V and the Cubic, ICMIC and Sinusoidal maps in the problem I. According to the results, the convergence time will decrease by 41.6–97.2%, on average, when using the Cubic, ICMIC and Gaussian maps.

In short, we can say that the Bernoulli Shift, Cubic, Sine, Singer, Tent, Circle, Gaussian, Intermittency, and Leibovitch maps have better performance than the standard LA in terms of the convergence rate and convergence time. In other words, the learning ability of the LA can be remarkably improved by using these chaotic maps in the learning cycle of the LA instead of the random action selection.

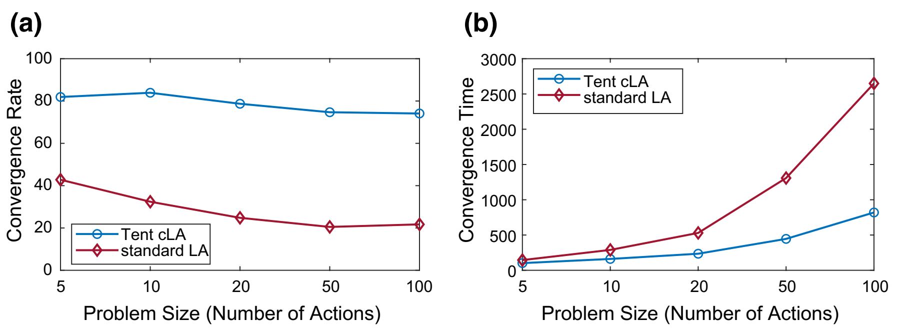


Fig. 7 Scalability comparison of Tent chaotic LA and standard LA. **a** Convergence rate versus problem size and **b** convergence time versus problem size

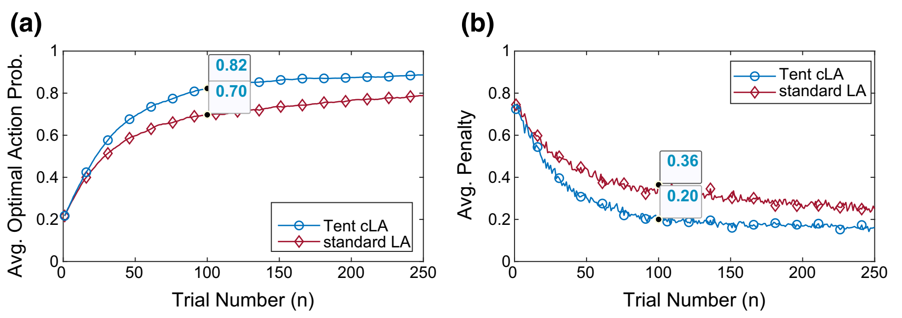


Fig. 8 Plots for problem P_{I}. **a** Average probability of the optimal action versus trial number and **b** average penalty versus the trial number

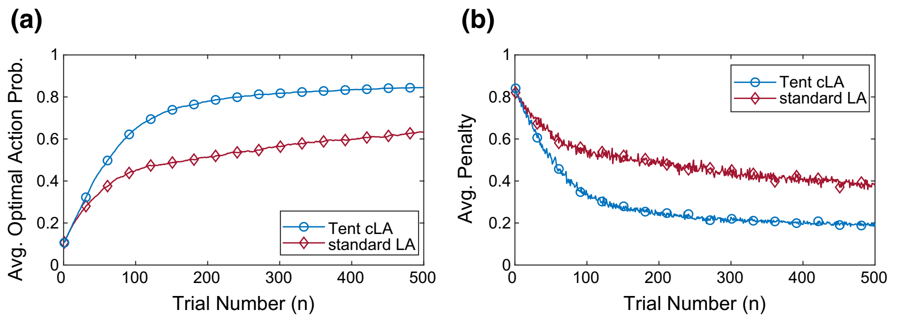


Fig. 9 Plots for problem P_{II}. **a** Average probability of the optimal action versus trial number and **b** average penalty versus the trial number

Furthermore, the Bernoulli Shift, Tent, and Leibovitch maps in terms of the convergence rate, and the Cubic, ICMIC, and Gaussian maps in terms of the convergence time ranked first to third. If we want to select a map from these maps, we must make

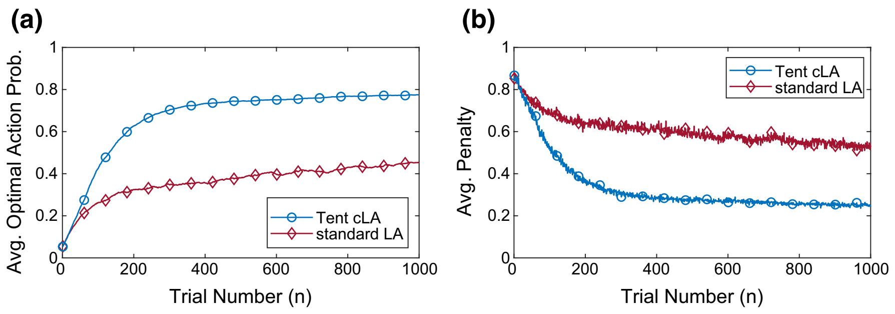


Fig. 10 Plots for problem P_{III}. **a** Average probability of the optimal action versus trial number and **b** average penalty versus the trial number

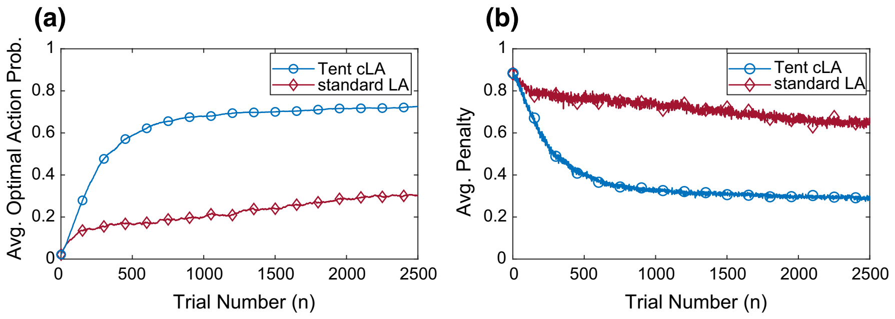


Fig. 11 Plots for problem P_{IV}. **a** Average probability of the optimal action versus trial number and **b** average penalty versus the trial number

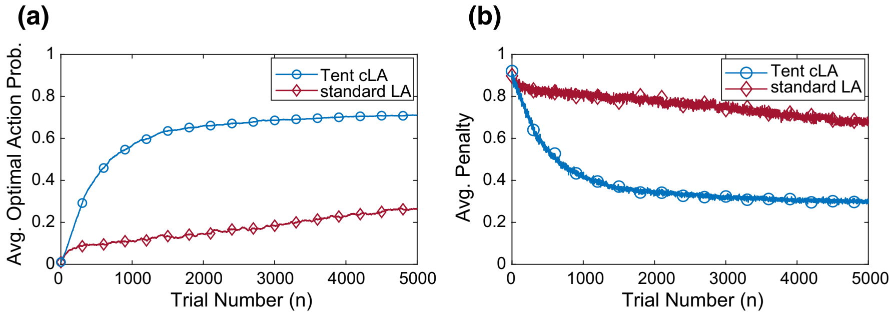


Fig. 12 Plots for problem P_V. **a** Average probability of the optimal action versus trial number and **b** average penalty versus the trial number

a compromise between these two criteria(convergence rate and convergence time). According to the results of Tables 2 and 3, we can say that the Tent map has better performance than the other maps in terms of these two criteria. By using the Tent map, the convergence rate/convergence time compared to the standard LA increases/

decreases by 91.4%/29.6% to 264.4%/69.1%, on average. Also, by increasing the problem size (number of actions) from 5 to 100, the convergence rate/convergence time in the standard LA decreases/increases by 49.3%/1730.8% and in the Tent map by 9.5%/704.7%, on average. It shows that the Tent chaotic LA has more scalability than the standard LA (see Fig. 7). The results of Tables 2 and 3 have been presented in “Appendix 1” as a chart for further analysis.

In Figs. 8, 9, 10, 11 and 12, the average probability of optimal action ($\hat{p}_{\mathrm{opt}}$) and average penalty ($\hat{M}$) of the Tent chaotic LA has been compared with standard LA. Figures 8a, 9a, 10a, 11a and 12a illustrate the $\hat{p}_{\mathrm{opt}}$ in the problems P_{I} to P_{V}, respectively, and Figs. 8b, 9b, 10b, 11b and 12b illustrate the $\hat{M}$ in the problems P_{I} to P_{V}, respectively. As you can see, the Tent chaotic LA has better performance than the standard LA in terms of both criteria, i.e., $\hat{p}_{\mathrm{opt}}$ and $\hat{M}$. For example, in the problem P_{I}, in the 100th trial, the Tent chaotic LA reaches to the $\hat{p}_{\mathrm{opt}} = 0.82$ and $\hat{M} = 0.20$ while the standard LA reaches to the $\hat{p}_{\mathrm{opt}} = 0.70$ and $\hat{M} = 0.36$ (see Fig. 8). As mentioned, the higher $\hat{p}_{\mathrm{opt}}$ and the lower $\hat{M}$ indicate better learning ability.

5.4 Shortcomings of the chaotic learning automaton (cLA)

According to the results of Sect. 5.3, it can be said that the use of chaotic numbers leads to a significant improvement in the learning ability of the LA. In general, the use of chaotic numbers has two major drawbacks. (1) to a large extent, it is not possible to say which chaotic map can be better suited for generating chaotic time series in a particular application. As you can see in Tables 2 and 3, in the chaotic LA, some of the chaotic maps have worse performance than the standard LA. Therefore, in the particular application, different chaotic maps need to be examined to determine the appropriate chaotic map(s). (2) Most chaotic maps require many computations to generate chaotic time series compared to the random number generators. This increases the required time to solve the problem. However, because the problem converges very quickly by using the chaotic time series, the computational overhead of the chaotic maps can be ignored.

6 Conclusion

In this paper, the chaos theory is incorporated with the LA and a new type of LA, namely chaotic LA (cLA), is introduced. In cLA, the chaotic numbers are used instead of the random numbers when choosing the action. The experiment results show that in most cases, the use of chaotic numbers leads to a significant improvement in the learning ability of the LA. Among the chaotic maps investigated in this paper, the Tent map has better performance than the other maps. The convergence rate/convergence time of the LA will increase/decrease by 91.4%/29.6% to 264.4%/69.1%, on average, by using the Tent map. Furthermore, the chaotic LA has more scalability than the standard LA, and its performance will not decrease significantly by increasing the problem size (number of actions). In this paper, the performance of the cLA with the linear learning algorithm is evaluated. Evaluating the performance of the cLA with nonlinear and hybrid learning algorithms can be the focus of our work in the future.

Appendix 1

In this appendix, the results of Tables 2 and 3 are presented as a chart for further analysis. In the charts of this appendix, the red horizontal dashed line indicates the convergence rate/convergence time of the standard LA, and the bars indicate the convergence rate/convergence time of the chaotic LAs. In the blue charts (Figs. 13, 14, 15, 16, 17), which illustrate the convergence rate, chaotic LAs whose bar are over the red horizontal dashed line have better performance than the standard LA. In the green charts (Figs. 18, 19, 20, 21, 22), which illustrate the convergence time, chaotic LAs whose bar are under the red horizontal dashed line have better performance than the standard LA. By investigating these charts, we can simply conclude that the Tent chaotic LA has better performance than the other chaotic LAs and standard LA in terms of both mentioned criteria, i.e., convergence rate and convergence time.

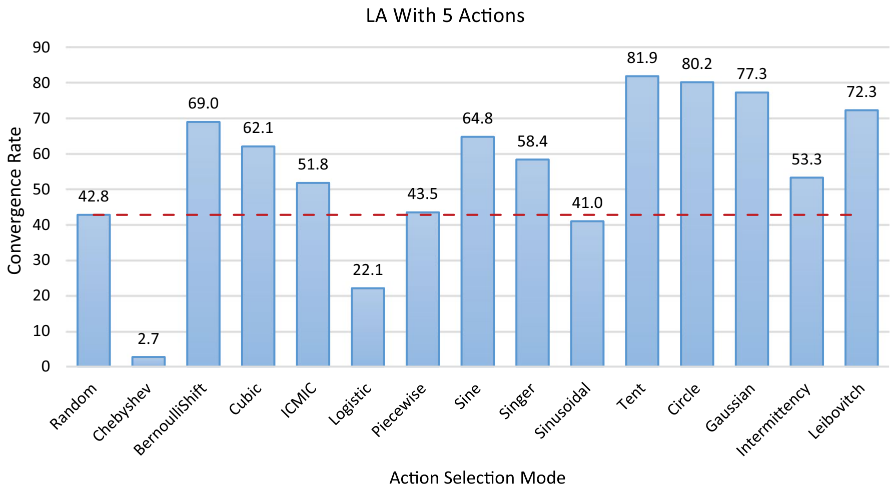


Fig. 13 Convergence rate in the problem P_{I} for different modes of action selection

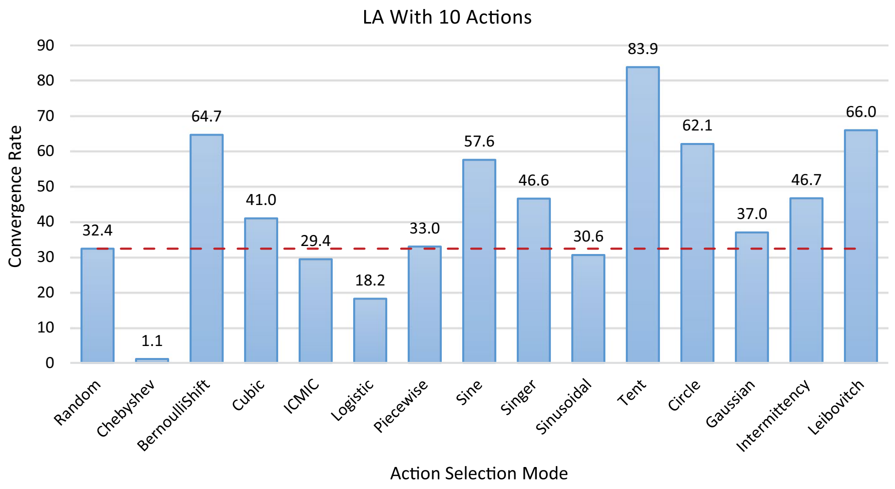


Fig. 14 Convergence rate in the problem P_{II} for different modes of action selection

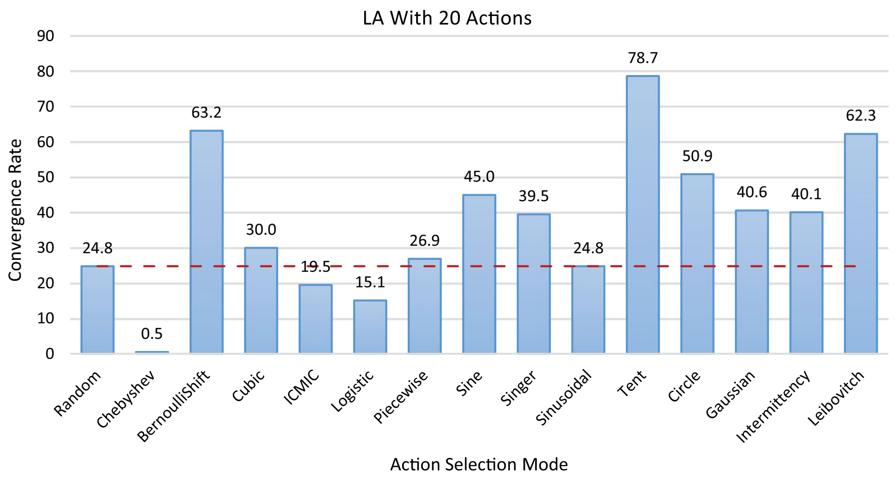


Fig. 15 Convergence rate in the problem P_{III} for different modes of action selection

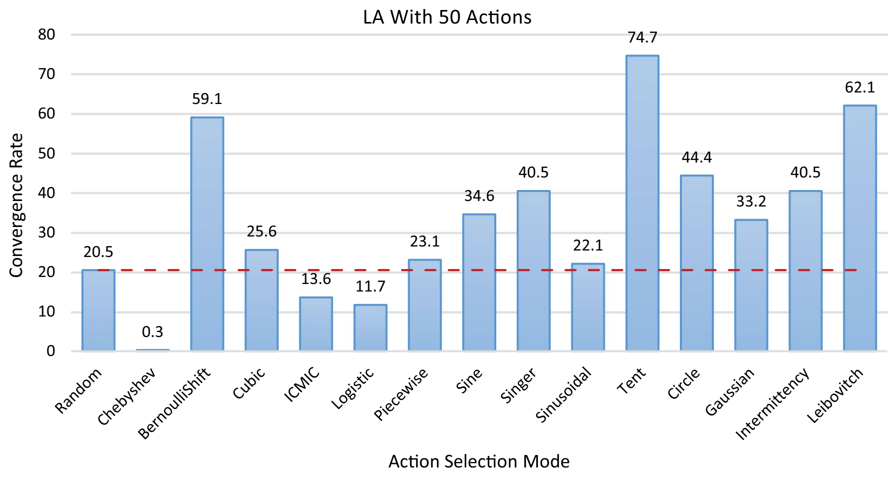


Fig. 16 Convergence rate in the problem P_{IV} for different modes of action selection

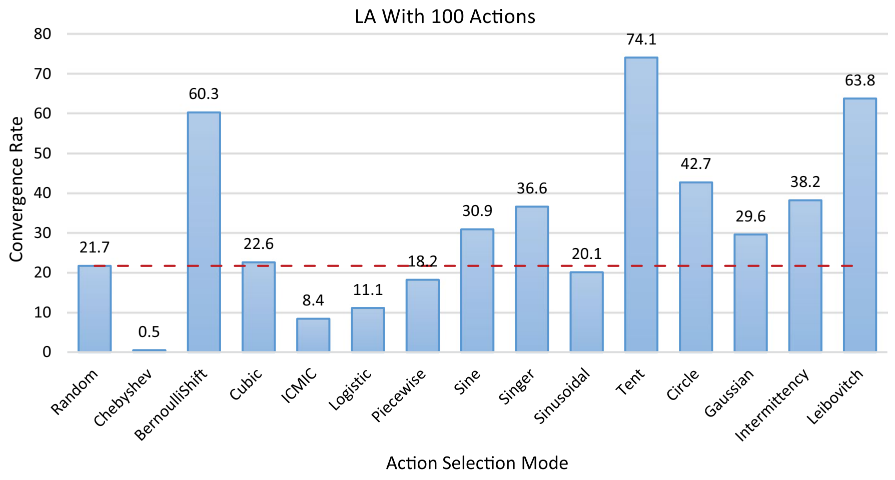


Fig. 17 Convergence rate in the problem P_{V} for different modes of action selection

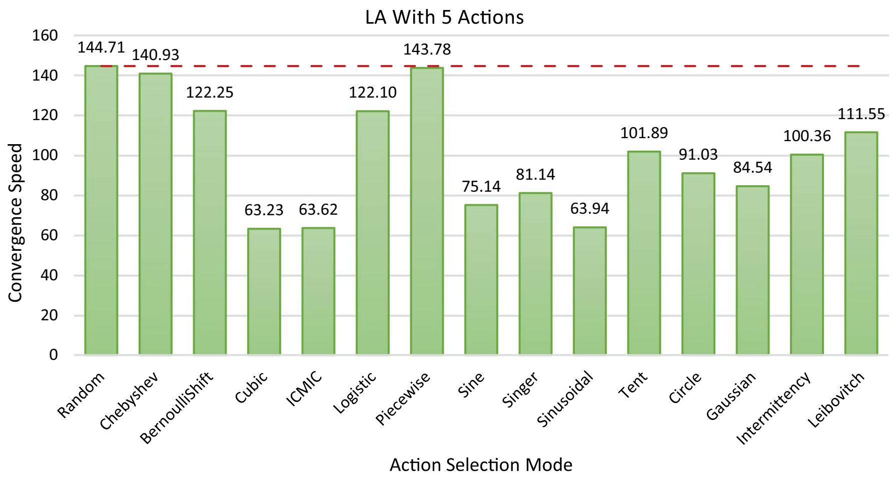


Fig. 18 Convergence time in the problem P_{I} for different modes of action selection

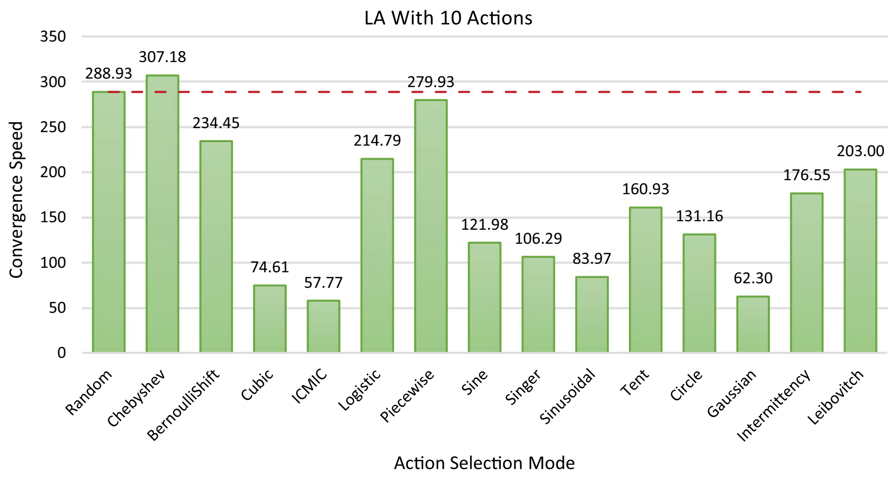


Fig. 19 Convergence time in the problem P_{II} for different modes of action selection

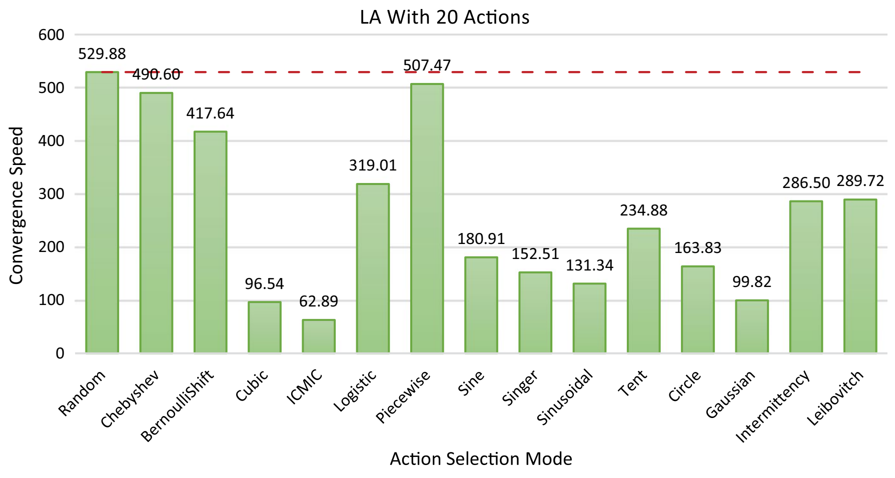


Fig. 20 Convergence time in the problem P_{III} for different modes of action selection

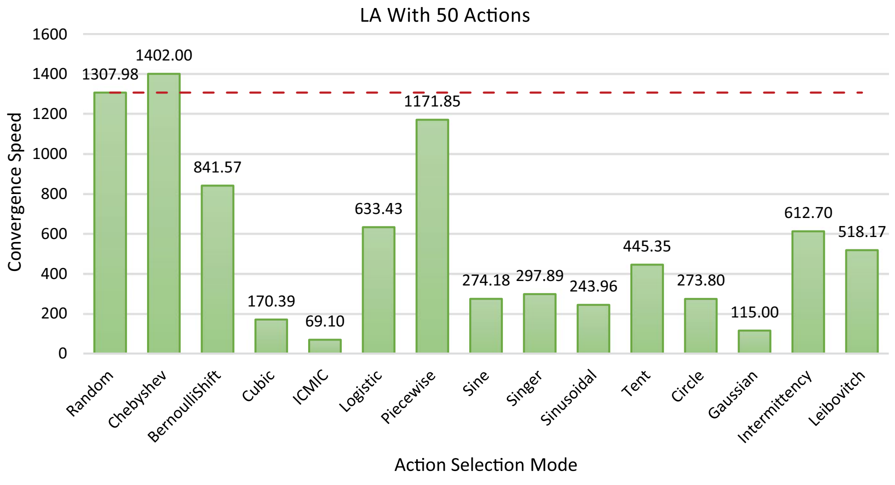


Fig. 21 Convergence time in the problem P_{IV} for different modes of action selection

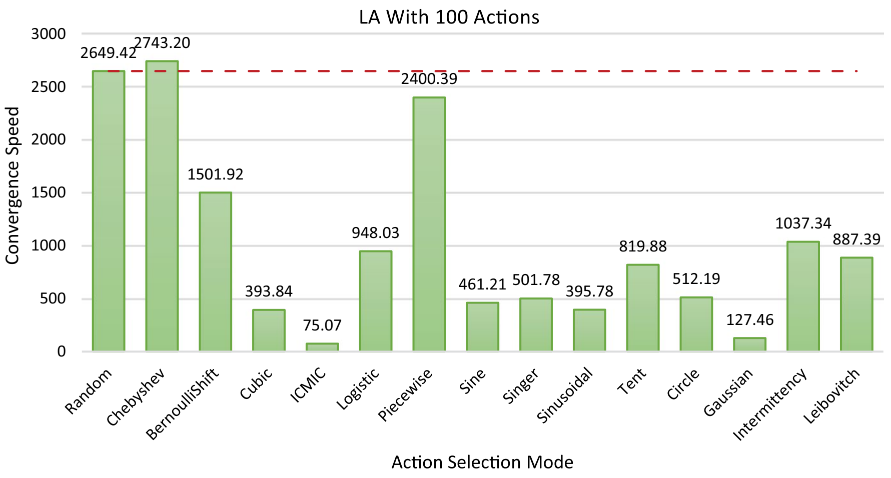


Fig. 22 Convergence time in the problem P_V for different modes of action selection

References

1. Bolouki Speily OR, Kardan A (2018) Modeling the information spreading in online blog communities using learning automata. Int J Web Res 1(2):43–55
2. Harmon ME, Harmon SS (1997) Reinforcement learning: a tutorial. WRIGHT LAB WRIGHT-PATTERSON AFB OH
3. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT press, Cambridge
4. Narendra KS, Thathachar MA (2012) Learning automata: an introduction. Courier Corporation, North Chelmsford
5. Caponetto R, Fortuna L, Fazzino S, Xibilia MG (2003) Chaotic sequences to improve the performance of evolutionary algorithms. IEEE Trans Evol Comput 7(3):289–304
6. Zarei B, Meybodi MR, Masoumi B (2020) Chaotic memetic algorithm and its application for detecting community structure in complex networks. Chaos Interdiscip J Nonlinear Sci 30(1):013125
7. Rezvanian A, Saghiri AM, Vahidipour SM, Esnaashari M, Meybodi MR (2018) Recent advances in learning automata. Springer, Berlin
8. Rezvanian A, Moradabadi B, Ghavipour M, Khomami MMD, Meybodi MR (2019) Learning automata approach for social networks. Springer, Berlin
9. Thathachar MA, Sastry PS (2011) Networks of learning automata: techniques for online stochastic optimization. Springer, Berlin
10. Rummery GA, Niranjan M (1994) On-line Q-learning using connectionist systems. University of Cambridge, Cambridge
11. Watkins CJCH (1989) Learning from delayed rewards. PhD thesis, Cambridge University
12. Barto AG, Sutton RS, Anderson CW (1983) Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans Syst Man Cybern 5:834–846
13. Schwartz A (1993) A reinforcement learning method for maximizing undiscounted rewards. In: Proceedings of the Tenth International Conference on Machine Learning, vol 298, pp 298–305
14. Lorenzelli F (2014) The essence of chaos. CRC Press, Boca Raton
15. Smith P (1998) Explaining chaos. Cambridge University Press, Cambridge
16. Williams G (1997) Chaos theory tamed. CRC Press, Boca Raton
17. Ausloos M, Dirickx M (2006) The logistic map and the route to chaos: from the beginnings to modern applications. Springer, Berlin
18. Hilborn RC (2000) Chaos and nonlinear dynamics: an introduction for scientists and engineers. Oxford University Press on Demand, Oxford
19. Gandomi AH, Yang X-S (2014) Chaotic bat algorithm. J Comput Sci 5(2):224–232

20. Jordehi AR (2015) A chaotic artificial immune system optimisation algorithm for solving global continuous optimisation problems. Neural Comput Appl 26(4):827–833
21. Lu H, Wang X, Fei Z, Qiu M (2014) The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms. Math Prob Eng 2014:1–16
22. Vorontsova I (1965) Algorithms for changing stochastic automata transition probabilities. Probl Peredachi Inform 1(3):122–126
23. Chandrasekharan B, Shen D (1968) On expediency and convergence in variable structure stochastic automata. IEEE Trans Syst Sci Cybern 5:145–149
24. Shapiro IJ, Narendra KS (1969) Use of stochastic automata for parameter self-optimization with multimodal performance criteria. IEEE Trans Syst Sci Cybern 5(4):352–360
25. Viswanathan R, Narendra KS (1972) A Note on the linear reinforcement scheme for variable-structure stochastic automata. IEEE Trans Syst Man Cybern SMC-2(2):292–294
26. Narendra KS, Thathachar MA (1974) Learning automata-a survey. IEEE Trans Syst Man Cybern 4:323–334
27. Meybodi M, Lakshmivarahan S (1982) ε-Optimality of a general class of learning algorithm. Inform Sci 28:1–20
28. Thathachar MA, Oommen BJ (1983) Learning automata processing ergodicity of the mean: the two-action case. IEEE Trans Syst Man Cybern 6:1143–1148
29. Poznyak S, Najim K (1997) On nonlinear reinforcement schemes. IEEE Trans Autom Control 42(7):1002–1004
30. Friedman EJ, Shenker S (1992) Learning by distributed automata. University of California, California
31. Thathachar MA, Sastry PS (1984) A class of rapidly converging algorithms for learning automata. In: IEEE International Conference on Cybernetics and Society, pp 602–606
32. Vasilakos AV, Papadimitriou GI (1995) A new approach to the design of reinforcement schemes for learning automata: stochastic estimator learning algorithm. Neurocomputing 7(3):275–297
33. Papadimitriou GI, Pomportsis AS, Kiritsi S, Talahoupi E (2001) Absorbing stochastic estimator learning algorithms with high accuracy and rapid convergence. In: Proceedings ACS/IEEE International Conference on Computer Systems and Applications. IEEE, pp 45–51
34. Lanctôt JK, Oommen BJ (1992) Discretized estimator learning automata. IEEE Trans Syst Man Cybern 22(6):1473–1483
35. Simha R, Kurose JF (1989) Relative reward strength algorithms for learning automata. IEEE Trans Syst Man Cybern 19(2):388–398
36. Vasilakos AV, Paximadis G (1994) Fault-tolerant routing algorithms using estimator discretized learning automata for high-speed packet-switched networks. IEEE Trans Reliab 43(4):582–593
37. Thathachar MA, Sastry PS (1986) Estimator algorithms for learning automata. In: Proceedings of the Platinum Jubilee Conference on Systems and Signal Processing, Department of Electrical Engineering, Indian Institute of Science, Bangalore, India
38. Oommen BJ, Agache M (2001) Continuous and discretized pursuit learning schemes: various algorithms and their comparison. IEEE Trans Syst Man Cybern Part B (Cybernet) 31(3):277–287
39. Oommen BJ, Lanctôt JK (1990) Discretized pursuit learning automata. IEEE Trans Syst Man Cybern 20(4):931–938
40. Agache M, Oommen BJ (2002) Generalized pursuit learning schemes: new families of continuous and discretized learning automata. IEEE Trans Syst Man Cybern Part B (Cybern) 32(6):738–749
41. Ge H, Li S, Li J, Ren X (2017) A parameter-free learning automaton scheme. arXiv:1711.10111