# Weight-Assignment Last-Position Elimination-Based Learning Automata

Haiwei An(✉), Chong Di, and Shenghong Li

School of Cyber Space Security, Shanghai Jiao Tong University, 800 Dong Chuan Road, 200240 Shanghai, China
anhaiweil995@sjtu.edu.cn

**Abstract.** Learning Automata (LA) is an adaptive decision-making unit under the reinforcement learning category. It can learn the randomness of the environment by interacting with it and adaptively adjust its behavior to maximize its long-term benefits from the environment. This learning behavior reflects the strong optimization ability of the learning automaton. Therefor LA has been applied in many fields. However, the commonly used estimators in previous LA algorithms have problems such as cold start, and the initialization process can also affect the performance of the estimator. So, in this paper, we improve these two weaknesses by changing the maximum likelihood estimator to a confidence interval estimator, using Bayesian initialization parameters and proposes a new update strategy. Our algorithm is named as weight-assignment last-position elimination-based learning automata (WLELA). Simulation experiments show that the algorithm has higher accuracy and has the fastest convergence speed than various classical algorithms.

**Keywords:** Learning automaton · Weight-assignment · Bayesian initialization · Confidence interval estimator

## 1 Introduction

Learning automaton can be seen as an adaptive decision-making unit, It can constantly interact with the random environment to adjust its choices to maximize the probability of being rewarded. The process of the LA interacting with the environment is shown in Fig. 1; [1]. At each moment t, an action $\alpha(t)$ will be chosen by LA to interact with the random environment and receives the environment feeds back $\beta(t)$, which can be either a reward or a penalty. Then, the automaton updates the state probability vector according to the received feedback. Because it's simple algorithm, strong anti-noise ability and strong optimization ability, it has received extensive attention and has been applied in many fields, such as random function optimization, QoS Optimization and certificate authentication.


Q. Liang et al. (Eds.): CSPS 2019, LNEE 571, pp. 348–354, 2020
https://doi.org/10.1007/978-981-13-9409-6_41

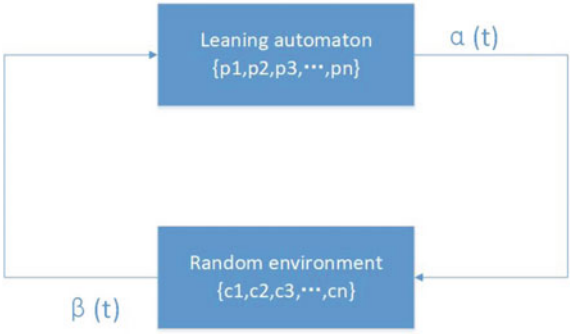


**Fig. 1.** LA interacts with the random environment

In the LA field, the most classical discrete pursuit algorithm with deterministic estimator is the DPRI algorithm given by Oommen in [2]. The main idea of the DPRI algorithm is to increase the probability vector which has the maximum value in the running estimates when the environment rewards the current action and decreases others; otherwise, the automaton changes nothing. Furthermore, many classical pursuit algorithms, such as DGPA [3] and SERI [4], also use estimators and discretization to improve the convergence speed of automaton. In [5], an algorithm named last-position elimination-based LA( LELA) which is contrary to the classical pursuit algorithm is proposed. Instead of greedily increasing the probability vector of the optimal estimation action, this algorithm reduces the probability of choosing the current worst estimation action, t Experiments show that LELA can get faster convergence speed than DGPA.

However, the estimator used by LELA has some innate defects. One typical flaw is the cold start and initialization problem, for example, since the maximum likelihood estimator does not have any information at the beginning, each action has to interact with the environment a certain number of times, it will Increase the cost of getting information in some complicated situations. And the update strategy of the LELA algorithm simply makes all active actions equally share the penalized state probability from the last-position action, does not consider the difference between optimal action and other actions at all. Thus, in this paper, we propose a weight assignment LELA (WLELA) algorithm which has made the following three changes: 1. Improvement of initialization parameters; 2. the estimator improvement; 3. changes in the probability vector update strategy.

In Sect. 2, a brief introduction of the LELA algorithm is given. In Sect. 3 we will show our algorithm WLELA in detail. Then in Sect. 4 part, we will give the simulation results of the WLELA algorithm, which will be compared with LELA and other classic algorithms such as DPRI, DGPA. Finally, summarize in Sect. 5.

## 2 Related Work

LA can be represented by a four-tuple ⟨A, B, Q, T⟩ model. They are explained as follows.

A is the set of actions. B is the feedbacks from random environment, when B = {0, 1}, where β = 0 represents that the LA has been penalized, and β = 1 means the LA has been rewarded. Q = <P, E>, E represents the estimator, it contains all the historical information that each action interacts with the environment. The most commonly used estimator is the maximum likelihood estimator. P is the state probability vector of choosing an action α(t) at any instant t, it satisfies ∑pi(t) = 1. T is the state transition function of LA, which determines how LA migrates to the state of t + 1 according to the state of output, input, and t at time t.

The random environment can also be described by a triple <A, B, C> mathematical model. where A and B are defined in the same way as above, and C is defined as C = {cij = Pr{β(t) = βj|α(t) = αi}}.

In the original LELA algorithm [5], it uses the maximum likelihood estimator to record the historical information of all actions, according to the following formula

$$\mathrm{di(t)} = \frac{Wi(t)}{Zi(t)} \tag{1}$$

where Zi(t) is the number of times the action ai was selected up to time instant t and Wi (t) is the sum of the environmental feedbacks received up to time t. when an action is rewarded, the automaton will select the worst performing action from the estimator vector set and decrease corresponding state probability vector by a step Δ = 1/rn, where r is the number of allowable actions and n is a resolution parameter. If some action's state probability vector is reduced to zero during the process, this action will be removed from the optional set of actions, while the remaining actions will evenly share the state probability value from each decrease. The update scheme is described as follows:

**If** β(t) = 1 **then**
Find m ∈ $\mathrm{N_r}$ such that

$$d_m(t) = \min\{d_i(t)|p_i(t) \neq 0\}, i \in N_r$$

$$p_m(t+1) = \max\{p_m(t) - \Delta, 0\}$$

**If** $p_m(t+1) = 0$ **Then** $\mathrm{k(t)} = k(t) - 1$ Endif

$$p_j(t+1) = \min\left\{p_j(t) + \frac{p_m(t) - p_m(t+1)}{k(t)}, 1\right\}, \forall j \in N_r, such\ that p_t(t) > 0.$$

**Else**

$$p_i(t+1) = p_i(t) \forall i \in N_r$$

**Endif**.

κ(t) denotes the number of active actions and is initialized by r.
LELA has been proved to be ε-optimal in every stationary random environment.

## 3 Proposed Learning Automata

In order to overcome the shortcomings of the LELA algorithms, we propose the following improvements.

Firstly, we use the Bayesian estimator introduced in [6] to solve the cold start and initialization problems. However, if the Bayesian estimator is used directly in the LA algorithm, the convergence speed will be additionally affected, so in WLELA, we directly modify it to the mean of the posterior distribution which is to set all actions' $d_i(0) = 0.5$, thereby improving convergence efficiency while ensuring overcoming cold start and initialization problems.

Secondly, in order to get more information, in the WLELA algorithm we used the confidence interval estimator proposed in [7] which is

$$d_i(t) = \left\{1 + \frac{Z_i(t) - W_i(t)}{(W_i(t)+1)F_{2(W_i(t)+1),2(Z_i(t)-W_i(t),0.005}}\right\}^{-1}, \forall i \in N_r \tag{2}$$

where $F_{2(W_i(t)+1),2(Z_i(t)-W_i(t),0.005}$ is the 0.005 right tail probability of the F distribution $2(W_i(t)+1)$ and $2(Z_i(t) - W_i(t)$ dimensional degrees of freedom.

Last, since all state probability vectors add up to a total of 1, so the value of each state probability vector can be thought of as its weights in the vector set. So, WLELA increase their probability vector according to their weights. In this way, similar to the idea of finding the optimal action in the classic pursuit algorithm, the probability vector of the optimal action will get more attention when updating, so that more values can be added to the optimal action's state probability vector each time. Assigning the added value by weight is more in line with the purpose of learning the automatic machine to select the optimal action.

A detailed description of the WLELA is as follows

**Algorithm WLELA**

**Initialize** $p_i(0) = \frac{1}{r}; W_i(0) = 1, Z_i(0) = 2, \forall i \in N_r$

**Initialize** $d_i(0) = \left\{1 + \frac{Z_i(0)-W_i(0)}{(W_i(0)+1)F_{2(W_i(0)+1),2(Z_i(0)-W_i(0),0.005}}\right\}^{-1}, \forall i \in N_r$

**Step 1**: At time t, pick $\alpha(t) = \alpha_i$ according to the state probability vector *P(t);*

**Step 2**: Receive feedback $\beta_i(t)$ {0,1}. Update the estimate values

$$W_i(t) = W_i(t-1) + \beta_i(t),$$

$$Z_i(t) = Z_i(t-1) + 1$$

$$d_i(t) = \left\{1 + \frac{Z_i(t) - W_i(t)}{(W_i(t)+1)F_{2(W_i(t)+1),2(Z_i(t)-W_i(t),0.005}}\right\}^{-1}$$

**Step 3**: **If** $\beta_i(t) = 1$ **Then**

Find $m \in N_r$ such that

$$d_m(t) = \min\{d_i(t) | p_i(t) \neq 0\}, i \in N_r$$

$$p_m(t+1) = \max\{p_m(t) - \Delta, 0\}$$

**If** $p_m(t+1) = 0$ Then k(t) $= k(t) - 1$ **Endif**

$$p_j(t+1) = \min_{\forall j \in N_{r, p_j(t)} > 0} \left\{ p_j(t) + \{p_m(t) - p_m(t+1)\} \times \{p_j(t) + \frac{p_m(t) - p_m(t+1)}{k(t)}\}, 1 \right\}$$

**Endif**
**Step 4**: **If** $\beta_i(t) = 0$ **Then**

$$p_i(t+1) = p_i(t) \forall i \in N_r$$

*Goto* Step 1.
**Endif**
**Step 5**: **If** $\max\{P(t)\} = 1$,
**Then** CONVERGE to the action whose $p = \max\{P(t)\}$.
**ELSE**
*Goto* step 1.
**Endif**
**END**
The parameter k(t) has the same meaning in algorithm LELA.

## 4 Simulation Results

This section we compare the relative performances of the proposed WLELA with the LELA and the classical pursuit algorithms DPRI and DGPA by presenting their accuracy and convergence speed. The random environment we used is the most commonly used benchmark environment E1-E4 with 10 allowable actions as shown in Table 1.

**Table 1.** Benchmark environments

| | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ | $C_9$ | $C_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| E1 | **0.60** | 0.50 | 0.45 | 0.40 | 0.35 | 0.30 | 0.25 | 0.20 | 0.15 | 0.10 |
| E2 | **0.55** | 0.50 | 0.45 | 0.40 | 0.35 | 0.30 | 0.25 | 0.20 | 0.15 | 0.10 |
| E3 | **0.70** | 0.50 | 0.30 | 0.20 | 0.40 | 0.50 | 0.40 | 0.30 | 0.50 | 0.20 |
| E4 | 0.10 | 0.45 | **0.84** | 0.76 | 0.20 | 0.40 | 0.60 | 0.60 | 0.50 | 0.30 |

In the process of the LA simulation experiment, if the state probability vector of a certain action exceeds the set threshold T($0 < T \leq 1$), the algorithm is considered to have converged, if the converged action has the highest reward probability in the environment, it is considered that the learning automaton converged correctly.

For all the algorithms in the experiment, they are simulated with their best parameters, which are defined as the values that yielded the fastest convergence speed and guaranteed the automaton converged to the optimal action in a sequence of NE experiments. Specifically, in our experiment, we set the same threshold T and NE in [2, 3, 5], that is, T = 0.999 and NE = 750. After adjusting to the best parameters, we carried out 250,000 experiments to evaluate the average convergence rate and accuracy.

Accuracy is an indicator for judging the performance of an automaton, accuracy is defined as the probability that a learning automaton converges to the optimal action in an environment. As can be seen from Table 2, "Res" denotes the best resolution parameter, all algorithms can converge with high accuracy, while WLELA has higher accuracy than other algorithms, although the difference is not insignificant.

**Table 2.** Accuracy (number of correct convergences/number of experiments)

| ENV | WLELA | | LELA | | DGPA | | $DP_{RI}$ | |
|---|---|---|---|---|---|---|---|---|
| | Res | Acc | Res | Acc | Res | Acc | Res | Acc |
| E1 | n = 24 | 0.997 | n = 20 | 0.996 | n = 65 | 0.996 | n = 653 | 0.994 |
| E2 | n = 98 | 0.997 | n = 68 | 0.995 | n = 204 | 0.995 | n = 3221 | 0.993 |
| E3 | n = 12 | 0.998 | n = 10 | 0.997 | n = 28 | 0.99 | n = 216 | 0.996 |
| E4 | n = 31 | 0.998 | n = 27 | 0.997 | n = 55 | 0.997 | n = 881 | 0.994 |

## A. Average converge times

Convergence speed is one of the most critical performance indicators in learning automata. Convergence speed comparison data is shown in Table 2, "Ite" denotes the convergence speed.

From the Table 3, we can see that the WLELA algorithm is better than other algorithms in terms of convergence speed. Compared with LELA, the rate of convergence improvement in each environment is {6.93, 19.76, 3.13, 12.49 %}. Compared with the traditional DGPA and DPRI algorithms, the rate of improvement is {27.91, 40.43, 18.01, 28.28%} and {51.45, 72.24, 21.65, 56.02%}. It can be seen that WLELA converges faster than the other three algorithms, and the E2 environment is the most complex compared to other environments, and WLELA still performs best.

**Table 3.** Convergence speed

| ENV | WLELA | | LELA | | DGPA | | $DP_{RI}$ | |
|---|---|---|---|---|---|---|---|---|
| | Res | Ite | Res | Ite | Res | Ite | Res | Ite |
| E1 | n = 24 | 1209 | n = 20 | 1299 | n = 65 | 1677 | n = 653 | 2490 |
| E2 | n = 98 | 3090 | n = 68 | 3851 | n = 204 | 5187 | n = 3221 | 11,132 |
| E3 | n = 12 | 619 | n = 10 | 639 | n = 28 | 755 | n = 216 | 790 |
| E4 | n = 31 | 1037 | n = 27 | 1185 | n = 55 | 1446 | n = 881 | 2358 |

## 5 Conclusion

This paper proposes an improved algorithm WLELA. By using Bayesian initialization eliminates the cold start problem, using confidence interval estimator gets more interactive information and using weight allocation strategy to realize the classical LA's idea of pursuing the best behavior. These three improvements allow the WLELA algorithm to achieve high accuracy and fast convergence in the simulation experiments, and the results show that WLELA not only has the highest accuracy, but also the fastest convergence speed. Especially in the most complex environment, the WLELA still performs very well. In future work, consider using a random estimator instead of a deterministic estimator in WLELA. and the WLELA algorithm can be used in many applications that need to learn automata.

**Acknowledgements.** This work was supported by the National Key Research and Development Project of China under Grant 2016YFB0801003.

## References

1. Thathachar M, Sastry PS (2004) Networks of learning automata: techniques for online stochastic optimization. Kluwer, Dordrecht
2. Oommen BJ, Lanctôt JK (1990) Discretized pursuit learning automata. IEEE Trans Syst Man Cybern 20(4):931–938
3. 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
4. Papadimitriou GI, Sklira M, Pomportsis AS (2004) A new class of ε-optimal learning automata. IEEE Trans Syst Man Cybern Part B 34(1):246–254
5. Zhang J, Wang C, Zhou MC (2014) Last-position elimination-based learning automata. IEEE Trans Cybern 44(12):2484–2492
6. Xuan Z, Granmo OC, Oommen BJ (2013) On incorporating the paradigms of discretization and Bayesian estimation to create a new family of pursuit learning automata. Appl Intell 39 (4):782–792
7. Hao G, Jiang W, Li S et al (2015) A novel estimator based learning automata algorithm. Appl Intell 42(2):262–275