Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of the Franklin Institute 343 (2006) 27–47

Journal of The Franklin Institute

www.elsevier.com/locate/jfranklin

A new continuous action-set learning automaton for function optimization[☆]

Hamid Beigy[a,b,*], M.R. Meybodi[b,c]

[a]Computer Engineering Department, Sharif University of Technology, Tehran, Iran
[b]Institute for Studies in Theoretical Physics and Mathematics (IPM), School of Computer Science, Tehran, Iran
[c]Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran

Abstract

In this paper, we study an adaptive random search method based on continuous action-set learning automaton for solving stochastic optimization problems in which only the noise-corrupted value of function at any chosen point in the parameter space is available. We first introduce a new continuous action-set learning automaton (CALA) and study its convergence properties. Then we give an algorithm for optimizing an unknown function.
© 2005 The Franklin Institute. Published by Elsevier Ltd. All rights reserved.

1. Introduction

Many engineering problems, such as adaptive control, pattern recognition, filtering, and identification, under the proper assumptions can be regarded as the parameter optimization problems. Consider a system with measurements $g(x, \alpha)$ on the performance function $M(\alpha) = E[g(x, \alpha)]$, where α is the parameter and x is the observation, the parameter optimization problem is to determine the optimal

[☆]This research was in part supported by a grant from Institute for Studies in Theoretical Physics and Mathematics (IPM) (No. CS1382-4-04), Tehran, Iran.

[*]Corresponding author. Computer Engineering Department, Sharif University of Technology, Tehran, Iran. Tel.: +98 2166005616; fax: +98 2166019246.

E-mail addresses: beigy@ce.sharif.edu (H. Beigy), meybodi@ce.aut.ac.ir (M.R. Meybodi).

0016-0032/$30.00 © 2005 The Franklin Institute. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.jfranklin.2005.07.004

parameter α^* such that the performance function is extremized. The performance function $M(.)$ may also be a multi-modal function. Many efficient methods like steepest descent method, Newton's method, are available when the gradient ∇M is explicitly available. Usually, due to the lack of sufficient information concerning the structure of the function M or because of mathematical intractability, the function M to be optimized is not explicitly known and only noise-corrupted value of the function $M(\alpha)$ at any chosen point α can be observed.

Two important classes of algorithms are available for solving the above problem when only the noise-corrupted observations are available: stochastic approximation based algorithms [1] and learning automata based algorithms [2,3]. Stochastic approximation algorithms are iterative algorithms in which the gradient of the function M is approximated by a finite difference method, and using the function evaluations obtained at points, which are chosen close to each other [1]. The learning automata are adaptive decision making devices that operate in unknown random environments and progressively improve their performance via a learning process. Since learning automata theorists study the optimization under uncertainty, the learning automata are very useful for optimization of multi-modal functions when the function is unknown and only noise-corrupted evaluations are available. In these algorithms, a probability density function, which is defined over the parameter space, is used for selecting the next point. The reinforcement signal and the learning algorithm are used by the learning automata (LA) for updating the probability density function at each stage. We want that this probability density function converges to some probability density function where the optimal value of α is chosen with probability as close as to the unity. The distinguishing feature of the learning is that the probability distribution of $g(x, \alpha)$ is unknown. Methods based on stochastic approximation algorithms and learning automata represent two distinct approaches to the learning problems. Though both approaches involve iterative procedures, updating at every stage is done in the parameter space in the first method, which may result in a local optimum, and in the probability space in the second method. The learning automata methods have two distinct advantages over the stochastic approximation algorithms. The first advantage is that the action space need not be a metric space because as in the stochastic approximation algorithms in which the new value of the parameter is to be chosen close to the previous value. The second advantage is that the methods based on learning automata lead to global optimization, because at every stage any element of the action set can be chosen. Learning automata are divided into two main groups: finite action-set learning automata (FALA) and continuous action-set learning automata (CALA) based on whether the action set is finite or continuous [4]. FALA has finite number of actions and has been studied extensively. For an r-action FALA, the action probability distribution is represented by an r-dimensional probability vector that is updated by the learning algorithm. In many applications there is need to have large number of actions. The learning automata with too large number of actions converge too slowly. In such applications CALA, whose actions are chosen from real line, are very useful. For CALA, the action probability distribution is represented by a continuous function and this function is updated by learning algorithm at any stage. In the theory of LA, several algorithms for learning optimal parameters have been developed for many discrete and continuous parameters [2,3,5–7]. Gullapalli proposed a generalized learning automaton with continuous action set, which uses the context input for selecting its actions and the reinforcement signal for updating its parameters [8] and showed that this learning automaton finds the optimal action for each context vector. Vasilakos et al. introduced a generalized learning automaton with continuous action-set and showed that this learning automaton finds the optimal action for each context vector [9]. In [10], a CALA is given and its convergence has been shown. This CALA is used for stochastic optimization and needs at most two function evaluations in each iteration, irrespective of the dimension of the parameter space. In [11], a team of FALA and CALA is also used for stochastic optimization. In [12], a CALA, which is called *continuous action reinforcement learning automata* (CARLA) is given for adaptive control, but no formal study is conducted to explain its behavior.

In this paper, we study an adaptive random search method for finding the global minimum of an unknown function. In the first part of the paper, we introduce a new continuous action-set learning automaton and study its convergence behavior. We state and prove a strong convergence theorem for this learning automaton. In the second part of the paper, we propose an algorithm, which uses the proposed CALA for stochastic optimization. The proposed algorithm is a constant step size learning algorithm and needs only one function evaluation in each stage. The proposed algorithm is independent of the dimension of the parameter space of the function to be optimized.

The rest of this paper is organized as follows: Section 2 presents a brief review of learning automata. In Section 3, a new continuous action-set learning automaton is given and its behavior is studied. In Section 4, an algorithm for stochastic optimization and the experiments are given and Section 5 concludes the paper.

2. Learning automata

Learning automata (LA) are adaptive decision making units that can learn to choose the optimal action from a set of actions by interaction with an unknown random environment. At each instant n, the LA chooses an action α_n from its action probability distribution and applies it to the random environment. The random environment provides a stochastic response, which is called reinforcement signal, to the LA. Then the LA uses the reinforcement signal and a learning algorithm to update the action probability distribution. Based on the nature of the reinforcement signal, the random environment could be classified into three classes: P-, Q-, and S-model environments. The reinforcement signal in P-model environments has two values while in Q-model environments can take a finite number of values in the interval [0,1]. In S-model environment, the reinforcement signal is a bounded continuous random variable. The LA can be classified into two main groups: FALA and CALA [4]. The action-set of FALA is finite, for example for an r-action ($2 \leqslant r < \infty$) FALA, the action probability distribution is represented by an r-dimensional probability vector and is updated by the learning algorithm. When the FALA is used for solving optimization problems, we need to discretize the parameter space, so that actions of LA can be possible values of the corresponding parameter. The accuracy of the solution is increased by choosing the finer discretization and hence increasing the number of actions of LA. However, increasing the number of actions of LA leads to slow convergence of the learning algorithm. In order to provide a higher rate of convergence, hierarchical structure LA [13], discretized LA [14], estimator algorithms [15–17], and pursuit algorithms [18–21] are introduced. A more satisfying solution would be to employ on LA model where the action-set can be continuous. Such model of LA is called continuous action-set learning automata in which the action set is the real line. Like FALA, the CALA also use a probability distribution function to choose an action and the learning algorithm updates this function based on the reinforcement signal.

In [10], a CALA is given, in which the action probability distribution at instant n is a normal distribution with mean μ_n and standard deviation σ_n. At each instant, the CALA updates its action probability distribution (based on its interaction with the environment) by updating μ_n and σ_n. Since the action set is continuous, instead of penalty probabilities for various actions, we now have a penalty probability function, M, defined by

$$M(\alpha) = E[\beta(\alpha_n)|\alpha_n = \alpha].$$

CALA has no knowledge of the penalty function $M(.)$. The objective of automaton is to identify the optimal action, which results in the minimum value of $M(.)$. This is to be achieved through the learning algorithm that updates the action probability distribution using the most recent interaction with the random environment. We shall denote the reinforcement signal in response to action α as $\beta_{(\alpha)}$ and thus

$$M(\alpha) = E[\beta(\alpha)].$$

The objective for CALA is to learn value of α at which $M(\alpha)$ attains a minimum. That is, we want the action probability distribution, $N(\mu_n, \sigma_n)$ to converge to $N(\alpha^*, 0)$ where α^* is a minimum of $M(\alpha)$. However, for some technical reasons (Eq. (2)), σ_n cannot converge to zero [10]. So, another parameter, $\sigma_{\mathrm{L}} > 0$, is used in which the objective of learning is to converge σ_n to σ_{L} and μ_n converging to α^*. By choosing σ_{L} sufficiently small, asymptotically CALA will choose actions sufficiently close corresponding to the minimum of M with probability sufficiently close to unity [10].

The learning algorithm for CALA is described below. Since the updating given for σ_n does not automatically guarantee that $\sigma_{n+1} \geqslant \sigma_n$, always a projected version of σ_n is used, denoted by $\phi[\sigma_n]$. Also, unlike FALA, this CALA interacts with the environment through choice of two actions at each instant. At each instant n, the CALA chooses $\alpha_n \in \Re$ at random from its current distribution $N(\mu_n, \sigma_n)$. Then it gets the reinforcement from the environment for the two actions: μ_n and α_n. Let these reinforcement signals be $\beta(\mu)$ and $\beta(\alpha)$. Then, the action probability distribution is updated as

$$\begin{aligned}\mu_{n+1} &= \mu_n + af_1[\mu_n, \sigma_n, \alpha_n, \beta(\alpha), \beta(\mu)],\\ \sigma_{n+1} &= \sigma_n + af_2[\mu_n, \sigma_n, \alpha_n, \beta(\alpha), \beta(\mu)] - C\alpha[\sigma_n - \sigma_{\mathrm{L}}],\end{aligned} \tag{1}$$

where $f_1(.), f_2(.)$, and $\phi(.)$ are defined as below:

$$\begin{aligned}f_1(\mu, \sigma, \alpha, \beta(\alpha), \beta(\mu)) &= \left[\frac{\beta(\alpha) - \beta(\mu)}{\phi(\sigma)}\right]\left[\frac{\alpha - \mu}{\phi(\sigma)}\right],\\ f_2(\mu, \sigma, \alpha, \beta(\alpha), \beta(\mu)) &= \left[\frac{\beta(\alpha) - \beta(\mu)}{\phi(\sigma)}\right]\left[\left(\frac{\alpha - \mu}{\phi(\sigma)}\right)^2 - 1\right],\\ \phi(\sigma) &= (\sigma - \sigma_{\mathrm{L}})I\{\sigma > \sigma_{\mathrm{L}}\} + \sigma_{\mathrm{L}},\end{aligned} \tag{2}$$

and $\sigma_{\mathrm{L}} > 0$, $C > 0$ and $a \in (0, 1)$ are parameters of the algorithm. For this algorithm it is shown that with arbitrarily large probability, μ_n will converge close to a minimum of $M(.)$ and σ_n will converge close to σ_{L}, if we choose a and σ_{L} sufficiently small and C sufficiently large [10].

Another continuous action-set learning automata called CARLA is given in [12,22], independently. This automaton can be described as follows. Let action-set of automaton be a bounded continuous random variable defined over the interval $[\alpha_{\min}, \alpha_{\max}] \in \Re$. The CARLA uses a continuous probability density function, $f(n)$ to choose its actions. It is assumed that no information about the actions is available at the start of learning and therefore the probabilities of actions are initially equal. Thus, the initial distribution is chosen as uniform distribution. The CARLA selects action α_n at instant n from distribution $f(.)$ and applies it to a S-model random environment that emits a response $\beta_n \in [0, 1]$. Based on the response β_n, the continuous probability density function $f(n)$ is updated according to the following rule:

$$f(n+1) = \begin{cases} a[f(n) + (1 - \beta_n)H(\alpha, \alpha_n)] & \text{if } \alpha_n \in [\alpha_{\max}, \alpha_{\min}],\\ 0 & \text{otherwise,}\end{cases} \tag{3}$$

where a is a normalization factor and $H(\alpha, r)$ is a symmetric Gaussian neighborhood function centered on $r = \alpha_n$ given by Eq. (4). The function $H(\alpha, r)$ has the effect of spreading the reward for neighborhood actions of the selected action.

$$H(\alpha, r) = \lambda \mathrm{e}^{-1/2(\alpha - r/\sigma)^2}, \tag{4}$$

where λ and σ are parameters that affect the height and the width of the neighborhood function. The asymptotic behavior of this continuous action learning automata is not studied.

Learning automata have been used successfully in many applications such as computer network [23–25], solving NP-Complete problems [26–28], capacity assignment [29,30], neural network engineering [31–34], and cellular networks [35–38] to mention a few.

3. A new continuous action-set learning automaton

In this section, we introduce a new CALA, which will be used later in Section 4 for stochastic optimization. For the proposed CALA, we use the Gaussian distribution, $N(\mu, \sigma)$ for selection of actions, which is completely specified by the first and second order moments, μ and σ. The learning algorithm updates the mean and the variance of the Gaussian distribution at any instant using the reinforcement signal β, obtained from the random environment. The reinforcement signal, $\beta \in [0, 1]$, is a noise-corrupted reinforcement signal, which indicates a noise-corrupted observation of function $M(.)$ at the selected action. The reinforcement signal β is a random variable whose distribution function coincides almost with the distribution $H(\beta|\alpha)$ that belongs to a family of distributions which depends on the parameter α. Let

$$\begin{aligned} M(\alpha) &= E[\beta(\alpha)|\alpha] \\ &= \int_{-\infty}^{\infty} \beta(\alpha)\, \mathrm{d}H(\beta|\alpha), \end{aligned}$$

be a penalty function with bound $\mathscr{M}$ corresponding to this family of distributions. We assume that $M(.)$ is measurable and continuously differentiable almost everywhere. The CALA has to minimize $M(.)$ by observing $\beta(\alpha)$. Let μ_n and σ_n be the mean and the standard deviation of the Gaussian distribution, respectively. Using the learning algorithm, we ideally want that $\mu_n \to \mu^*$ and $\sigma_n \to 0$ as time tends to infinity.

The interaction between the CALA and the random environment takes place as iterations of the following operations. Iteration n begins by selection of an action α_n by the CALA. This action is generated as a random variable from the Gaussian distribution with parameters μ_n and σ_n. The selected action is applied to the random environment and the learning automaton receives an evaluative signal $\beta(\alpha_n)$, which has the mean value $M(\alpha_n)$, from the environment. Then the learning automaton updates the parameters μ_n and σ_n. Initially $M(.)$ is not known and it is desirable that with interaction of learning automaton and the random environment, choose μ_n and σ_n converges to their optimal values which results the minimum value of $M(.)$. The learning automaton uses the following rule to update the parameters μ_n and σ_n.

$$\begin{aligned} \mu_{n+1} &= \mu_n - a\beta(\alpha_n)\sigma_n(\alpha_n - \mu_n), \\ \sigma_{n+1} &= f(\sigma_n), \end{aligned} \tag{5}$$

where a is learning rate and $f(.)$ is a function that produces a random sequence of σ_n (described later). Eq. (5) can be written as

$$\mu_{n+1} = \mu_n - a\sigma_n^2 y_n(\alpha_n), \tag{6}$$

where

$$y_n(\alpha_n) = \beta(\alpha_n)\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right). \tag{7}$$

An intuitive explanation for the above updating equations is as follows: we can view the fraction in Eq. (7) as the normalized noise added to the mean. Since a, μ_n and σ_n all are positive, the updating equation changes the mean value in the opposite direction of the noise. If the noise is positive, then the learning automaton should update its parameter, so that mean value increased and vice versa. Since $E[\beta|\alpha]$ is close to unity when α is far from its optimal value and it is close to zero when α is near to the optimal value, thus the learning automaton updates μ with large steps when α is far from its optimal value and with small steps when α is close to its optimal value. This causes a finer quantization of μ near its optimal value and a grain quantization for points far away its optimal value. Thus, we can consider the learning algorithm as a random direction search algorithm with adaptive step sizes.

In what follows, we state and prove the convergence of the proposed learning automaton in stationary environments. The convergence is proved based on the following assumptions:

Assumption 1. The sequence of real numbers $\{\sigma_n\}$ is such that

$$\sigma_n \geqslant 0,$$
$$\sum_{n=1}^{\infty} \sigma_n^3 = \infty,$$
$$\sum_{n=1}^{\infty} \sigma_n^4 < \infty.$$

Note that these conditions imply that $\sigma_n \to 0$ as $n \to \infty$. Therefore, in limit, σ_n of Gaussian distribution tends to zero and the action of learning automaton becomes equal to the mean. The condition $\sum_{n=1}^{\infty} \sigma_n^3 = \infty$ ensures that the sum of increments to the initial mean, μ_0, can be arbitrarily large, so that any finite initial value of μ_0 can be transformed into the optimal value μ^*. At the same time, the condition that $\sum_{n=1}^{\infty} \sigma_n^4 < \infty$ ensures that the variance in μ_n is finite and the mean cannot diverge to infinity.

Here σ_n^2 is like the step size, thus the above assumption is the standard assumption in stochastic approximation algorithms on the step size.

Assumption 2. There is a unique real number μ^* such that

$$M(\mu^*) = \min_{\alpha}(M(\alpha)).$$

This assumption implies that $M(.)$ has unique minimum. Let $\mathscr{M}$ be the bound on $M(\alpha)$. It is also assumed that $M(\alpha)$ has a finite number of minima inside a compact set and has bounded first and second derivatives with respect to α. Let

$$R(\alpha) = \frac{\partial M(\alpha)}{\partial \alpha}, \tag{8}$$

and

$$S(\alpha) = \frac{\partial M^2(\alpha)}{\partial \alpha^2}, \tag{9}$$

be the first and the second derivative of $M(\alpha)$, respectively. Suppose that $\mathscr{R}$ and $\mathscr{S}$ be the bounds on R and S, respectively. Note that this assumption ensures that there is an optimal action α^* for the learning automaton for which $E[\beta(\alpha^*)]$ is minimum. Since in limit $\sigma_n \to 0$, then this assumption ensures that there is an optimal mean μ^* for which M is minimized.

Assumption 3. $M(\alpha)$ is linear near μ^*, that is

$$\sup_{\varepsilon \leqslant |\alpha-\mu^*| \leqslant \frac{1}{\alpha}} (\alpha - \mu^*) R(\alpha) > 0, \tag{10}$$

for all $\varepsilon > 0$. This assumption is restrictive, because it requires $M(.)$ to be linear in the neighborhood of the optimal action.

Assumptions 2 and 3 mean that the function being optimized has a unique minimum and it behaves like as a quadratic in the search area and in near of the optimum point.

Assumption 4. The noise in the reinforcement signal $\beta(.)$ has a bounded variance, that is

$$\begin{aligned} E\{[\beta(\alpha) - M(\alpha)]^2\} &= \int_{-\infty}^{\infty} [\beta(\alpha) - M(\alpha)]^2 \, \mathrm{d}H(\beta|\alpha), \\ &\leqslant K_1[1 + (\alpha - \mu^*)^2], \end{aligned} \tag{11}$$

for some real number $K_1 > 0$. The above equation can be written as

$$E\{\beta^2(\alpha)\} \leqslant K_1[1 + (\alpha - \mu^*)^2] + M^2(\alpha).$$

Thus $E\{\beta^2(\alpha)\}$ is bounded by a quadratic function of α for all α.

Given the above assumptions, in what follows, we study the behavior of the proposed CALA. The following theorem states the convergence of the random process defined by (5). The method used to prove this theorem is similar to the methods used in stochastic approximation [1].

Theorem 1. *Suppose that the Assumptions* 1–4 *hold,* μ_0 *is finite and there is an optimal value of* μ^* *for* μ. *Then if* μ_n *and* σ_n *are evolved according to the given learning algorithm, then* $\lim_{n\to\infty} \mu_n = \mu^*$ *with probability* 1.

Before we prove the above theorem, we first prove the following lemmas:

Lemma 1.

$$E[y_n(\alpha_n)|\mu_n] = \sigma_n R(\mu_n).$$

Proof. Let β_n to denote $\beta(\alpha_n)$, where α_n is a random variable chosen from Gaussian distribution with mean μ_n and variance σ_n^2.

$$\begin{aligned} E[y_n(\alpha_n)|\mu_n] &= E\left[\beta_n\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)\middle|\mu_n\right] \\ &= E\left[M(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)\middle|\mu_n\right]. \end{aligned} \tag{12}$$

Eq. (12) is obtained through the total expectation law as given below [39].

$$E[X] = E[E[X|Y|]]. \tag{13}$$

Replacing $M(\alpha_n)$ with its second order Taylor series expansion around μ_n, we obtain

$$\begin{aligned} E[y_n(\alpha_n)|\mu_n] &= E\left[M(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)\middle|\mu_n\right] \\ &= E\left\{\left[M(\mu_n)+R(\mu_n)(\alpha_n-\mu_n)+S(\xi)\frac{(\alpha_n-\mu_n)^2}{2}\right]\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)\middle|\mu_n\right\} \\ &= M(\mu_n)E\left[\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)\middle|\mu_n\right] + R(\mu_n)E\left[\left(\frac{(\alpha_n-\mu_n)^2}{\sigma_n}\right)\middle|\mu_n\right] \\ &\quad + \frac{1}{2}E\left[S(\xi)\left(\frac{(\alpha_n-\mu_n)^3}{\sigma_n}\right)\middle|\mu_n\right], \end{aligned} \tag{14}$$

where ξ lies between α_n and μ_n. The first and last terms on the right-hand side of the above equation is zero because the odd moments of a Gaussian random variable, α_n, are zero. Since $M(.)$ has bounded second derivative, thus the above equation can be written as

$$\begin{aligned} E[y_n(\mu_n)|\mu_n] &= R(\mu_n)E\left[\left(\frac{(\alpha_n-\mu_n)^2}{\sigma_n}\right)\middle|\mu_n\right] \\ &= \sigma_n R(\mu_n). \qquad \square \end{aligned} \tag{15}$$

Lemma 2.

$$E[y_n^2(\alpha_n)|\mu_n] \leqslant K_2[1+(\mu_n-\mu^*)^2].$$

Proof.

$$\begin{aligned} E[y_n^2(\alpha_n)|\mu_n] &= \int_{-\infty}^{\infty} E\left\{\left[\beta_n\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)\right]^2\middle|\mu_n\right\} \mathrm{d}N(\alpha_n|\mu_n,\sigma_n) \\ &= \int_{-\infty}^{\infty} E\{\beta_n^2|\mu_n\}\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n). \end{aligned} \tag{16}$$

Since it is assumed that the noise in the reinforcement signal has a bounded variance and $\beta^2(\alpha)$ is bounded by a quadratic function for all α (Eq. (11)), thus we have

$$\begin{aligned}E\{\beta_n^2|\mu_n\} &= \int_{-\infty}^{\infty} \beta^2(\alpha)\,\mathrm{d}H(\beta|\alpha)\\ &\leqslant K_1[1+(\alpha-\mu^*)^2]+M^2(\alpha).\end{aligned}$$

By substituting the above equation in Eq. (16), we obtain

$$\begin{aligned}E[y_n^2(\alpha_n)|\mu_n] &= \int_{-\infty}^{\infty} E\{\beta_n^2|\mu_n\}\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\\
&\leqslant K_1\int_{-\infty}^{\infty}\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\\
&\quad + K_1\int_{-\infty}^{\infty}(\alpha_n-\mu^*)^2\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\\
&\quad + K_1\int_{-\infty}^{\infty}M^2(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\\
&= K_1\int_{-\infty}^{\infty}\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\\
&\quad + K_1\int_{-\infty}^{\infty}[(\alpha_n-\mu_n)^2+2(\alpha_n-\mu_n)(\mu_n-\mu^*)\\
&\quad + (\mu_n-\mu^*)^2]\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\\
&\quad + K_1\int_{-\infty}^{\infty}M^2(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n).\end{aligned}$$

Since the odd moments of a Gaussian random variable are zero, the above equation can be written as

$$\begin{aligned}E[y_n^2(\alpha_n)|\mu_n] &\leqslant K_1+3K_1\sigma_n^2+K_1(\mu_n-\mu^*)^2\\
&\quad + K_1\int_{-\infty}^{\infty}M^2(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n).\end{aligned}\tag{17}$$

Replacing $M(\alpha_n)$ with its second order Taylor series expansion around μ_n, the last term of the above equation equals to

$$\begin{aligned}&\int_{-\infty}^{\infty}M^2(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\\
&\quad=\int_{-\infty}^{\infty}\left[M(\mu_n)+R(\mu_n)(\alpha_n-\mu_n)+S(\xi)\,\frac{(\alpha_n-\mu_n)^2}{2}\right]^2\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n,\sigma_n)\end{aligned}$$

$$\leqslant \int_{-\infty}^{\infty} \left[\mathscr{M} + \mathscr{R}(\alpha_n - \mu_n) + \mathscr{S}\, \frac{(\alpha_n - \mu_n)^2}{2}\right]^2 \left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n), \qquad (18)$$

where ξ lies between α_n and μ_n and $\mathscr{R}$ and $\mathscr{S}$ are bounds of the first and the second derivative of $M(.)$, respectively. Thus the above inequality can be written as

$$\begin{aligned}
&\int_{-\infty}^{\infty} M^2(\alpha_n)\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\quad\leqslant \int_{-\infty}^{\infty} [\mathscr{M}^2 + 2\mathscr{M}\mathscr{R}(\alpha_n - \mu_n)]\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\qquad + \int_{-\infty}^{\infty} \mathscr{M}\mathscr{S}(\alpha_n - \mu_n)^2\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\qquad + \int_{-\infty}^{\infty} \mathscr{S}^2\, \frac{(\alpha_n - \mu_n)^4}{4}\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\qquad + \int_{-\infty}^{\infty} \mathscr{R}^2(\alpha_n - \mu_n)^2\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\qquad + \int_{-\infty}^{\infty} \mathscr{R}\mathscr{S}(\alpha_n - \mu_n)^3\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n).
\end{aligned}$$

Since the odd moments of a Gaussian random variable are zero, the above inequality can be written as

$$\begin{aligned}
&\int_{-\infty}^{\infty} M^2(\alpha_n)\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\quad\leqslant \mathscr{M}^2 \int_{-\infty}^{\infty} \left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\qquad + \mathscr{R}^2 \int_{-\infty}^{\infty} \frac{(\alpha_n - \mu_n)^4}{\sigma_n^2} \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\qquad + \mathscr{M}\mathscr{S} \int_{-\infty}^{\infty} \frac{(\alpha_n - \mu_n)^4}{\sigma_n^2} \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\qquad + \frac{\mathscr{S}^2}{4} \int_{-\infty}^{\infty} \frac{(\alpha_n - \mu_n)^6}{\sigma_n^2} \mathrm{d}N(\alpha_n|\mu_n, \sigma_n) \\
&\quad = \mathscr{M}^2 + 3\mathscr{R}^2\sigma_n^2 + 3\mathscr{M}\mathscr{S}\sigma_n^2 + \frac{15}{4}\, \mathscr{S}^2\sigma_n^4.
\end{aligned}$$

Substituting the above equation in Eq. (17), we obtain

$$\begin{aligned}
E[y_n^2(\alpha_n)|\mu_n] \leqslant{}& K_1 + 3K_1\sigma_n^2 + K_1(\mu_n - \mu^*)^2 + K_1\mathscr{M}^2 + 3K_1\mathscr{R}^2\sigma_n^2 \\
&+ 3K_1\mathscr{M}\mathscr{S}\sigma_n^2 + \tfrac{15}{4}K_1\mathscr{S}^2\sigma_n^4.
\end{aligned}$$

Since σ_n, $\mathscr{M}$, $\mathscr{R}$ and $\mathscr{S}$ are bounded random variables, there is a constant $K_2 > 0$ such that the above inequality can be written as

$$E[y_n^2(\alpha_n)|\mu_n] \leqslant K_2(1 + (\mu_n - \mu^*)^2). \qquad \square \tag{19}$$

Proof of Theorem 1. Let

$$e_n = \mu_n - \mu^*. \tag{20}$$

Then using Eq. (6), the following recursive formula can be written for e_n:

$$e_{n+1} = e_n - a\sigma_n^2 y_n(\alpha_n). \tag{21}$$

Squaring, taking conditional expectation given $\mu_0, \ldots, \mu_n$ both sides of the above equation and then using Lemmas 1 and 2, we obtain

$$\begin{aligned} E[e_{n+1}^2|\mu_0, \ldots, \mu_n] &= E[(e_n - a\sigma_n^2 y_n(\alpha_n))^2|\mu_0, \ldots, \mu_n] \\ &\leqslant e_n^2 + a^2 K_2 \sigma_n^4 (1 + e_n^2) - 2a^2 K_2 \sigma_n^3 e_n R(\mu_n) \\ &\leqslant e_n^2 + a^2 K_2 \sigma_n^4 (1 + e_n^2) \\ &= e_n^2 (1 + a^2 K_2 \sigma_n^4) + a^2 K_2 \sigma_n^4. \end{aligned} \tag{22}$$

Let

$$Z_n = e_n^2 \prod_{j=n}^{\infty} (1 + a^2 K_2 \sigma_j^4) + a^2 K_2 \sum_{j=n}^{\infty} \sigma_j^4 \prod_{i=j+1}^{\infty} (1 + a^2 K_2 \sigma_i^4). \tag{23}$$

Then using Eqs. (22) and (23), it is easy to show that

$$e_n^2 \leqslant Z_n, \tag{24}$$

and

$$E\{Z_{n+1}|\mu_0, \ldots, \mu_n\} \leqslant Z_n. \tag{25}$$

Taking conditional expectations given $Z_1, \ldots, Z_n$ on both sides of the above inequality, we obtain

$$E\{Z_{n+1}|Z_0, \ldots, Z_n\} \leqslant Z_n, \tag{26}$$

which shows that Z_n is a non-negative super-martingale. Thus we have

$$E\{Z_{n+1}\} \leqslant E\{Z_n\} \leqslant \cdots \leqslant E\{Z_1\} \leqslant \infty. \tag{27}$$

Therefore, using the martingale convergence theorems [40], Z_n converges with probability 1. Since $e_n^2 \leqslant Z_n$, hence we conclude that e_n^2 converges to η with probability 1, where $\eta < \infty$ is a random variable. Taking expectation on both sides of (22), we obtain

$$E[e_{n+1}^2] - E[e_n^2] \leqslant a^2 K_2 \sigma_n^4 (1 + E[e_n^2]) - 2a^2 K_2 \sigma_n^3 e_n R(\mu_n).$$

Adding the first n of these inequalities, we get

$$E[e_{n+1}^2] - E[e_1^2] \leqslant a^2 K_2 \sum_{j=1}^{n} \sigma_n^4 (1 + E[e_j^2]) - 2a^2 K_2 \sum_{j=1}^{n} \sigma_j^3 e_j R(\mu_j).$$

Adding $E[e_1^2]$ to both sides of the above inequality, we obtain

$$E[e_{n+1}^2] \leqslant E[e_1^2] + a^2 K_2 \sum_{j=1}^{n} \sigma_n^4 (1 + E[e_j^2]) - 2a^2 K_2 \sum_{j=1}^{n} \sigma_j^3 e_j R(\mu_j).$$

Since $E[e_{n+1}^2]$ is positive, the above inequality becomes

$$E[e_1^2] + a^2 K_2 \sum_{j=1}^{n} \sigma_n^4 (1 + E[e_j^2]) - 2a^2 K_2 \sum_{j=1}^{n} \sigma_j^3 e_j R(\mu_j) \geqslant 0.$$

From the above inequality, using the boundness of $E[e_j^2]$ (for $j > 0$), and Assumption 1, it follows that

$$2a^2 K_2 \sum_{j=1}^{n} \sigma_j^3 e_j R(\mu_j) \leqslant E[e_1^2] + \sum_{j=1}^{n} a^2 K_2 \sigma_n^4 (1 + E[e_j^2])$$
$$< \infty.$$

Since $\sum_{j=1}^{\infty} \sigma_j^3$ diverges and by Assumption 3, the quantity of $e_j R(\mu_j)$ is positive, we can conclude that for some sequence $\{n_j\}$, we have

$$e_{nj} R(\mu_{nj}) \to 0, \tag{28}$$

with probability 1. The fact that e_n^2 converges with probability 1 to some random variable η together with Assumptions 1–4 and the above equation imply that $\eta = 0$ with probability 1. Hence μ_n converges to μ^* with probability 1.

4. Optimization of a function

In this section, we give an algorithm based on the continuous action-set learning automaton given in Section 3, for optimization of an unknown function. The proposed algorithm can be easily extended to multivariate functions by using a cooperative game of CALA with identical payoff. Consider function, $F : \Re \to \Re$ to be minimized. At any instant n, the automaton chooses an action α_n using the Gaussian distribution $N(\mu_n, \sigma_n)$, where μ_n and σ_n is the mean and the standard deviation, respectively. As before, let $\beta(\alpha)$ denote the noisy function evaluation at point $\alpha \in \Re$ such that $\beta(\alpha) = (f(\alpha) - \lambda_2)/\lambda_1)$, where λ_1 and λ_2 are two appropriate scaling constant such that $\beta(a) \in [0, 1]$ and $f(.)$ is noisy evaluation of $F(a)$. This is because of the assumption that β is a bounded and non-negative random variable. The signal $\beta(\alpha)$ is used to update the μ of the Gaussian distribution (Eq. (5)).

In order to study the behavior of the proposed algorithm, we consider the optimization of the penalized Shubert function, which is borrowed from [10], when the function evaluations are noisy. We assume that only the function evaluations are available. The penalized Shubert function is given below.

$$F(x) = \sum_{i=1}^{5} i \cos((i+1)x + 1) + u(x, 10, 100, 2), \tag{29}$$

where u is the penalizing function given by

$$u(x, b, k, m) = \begin{cases} k(x-b)^m, & x > b, \\ 0, & |x| \leqslant b, \\ k(-x-b)^m, & x < -b. \end{cases} \tag{30}$$

This function has 19 minima within interval $[-10, 10]$ where three of them are global minima. The global minimum value of the penalized Shubert function is approximately equal to -12.87 and is attained at points close to $-5.9, 0.4$, and 6.8. The penalized Shubert function is frequently used as one of the benchmark problems for global optimization problems (Fig. 1).

The proposed continuous action-set learning automaton and the automaton proposed in [10] with different initial values of the mean and variance parameters are used for finding the minimum of penalized Shubert function. The algorithm given in [10] is used with the following values of parameters: the penalizing constant C is equal to 5, the lower bound for the variance, $\sigma_l = 0.01$ and the step size

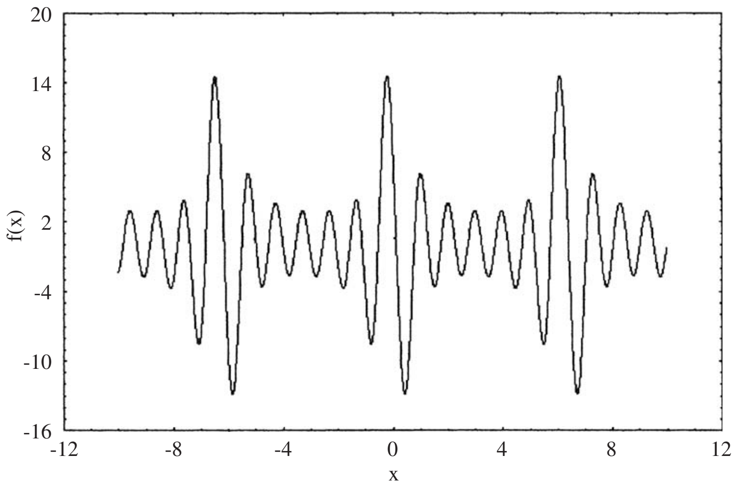

Fig. 1. The penalized Shubert function.

Table 1
The simulation results with white noise $U[-0.5, 0.5]$ added to the penalized Shubert function evaluations

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.716046	0.1	−12.861304	2.534	0.01	−3.578
2	4	10	0.436827	0.1	−12.848987	0.4308	0.01	−12.87
3	8	5	7.805426	0.1	−3.5772	5.364	0.01	−8.5
4	8	3	6.704283	0.1	−12.868272	6.72	0.01	−12.87
5	12	6	5.460669	0.1	−8.476369	1.454	0.01	−3.58
6	−10	5	−8.107043	0.1	−3.749682	−7.1	0.01	−8.5
7	−10	6	−7.09686	0.1	−8.507154	−5.8	0.01	−12.87

Table 2
The simulation results with white noise $N(0, 0.1)$ added to the penalized Shubert function evaluations

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.725683	0.107958	−12.82271	6.735791	2.204917	−12.75081
2	4	10	6.727295	0.107958	−12.813384	9.213571	0.624627	2.763267
3	8	5	−5.84428	0.107958	−12.840563	10	0.038636	−0.002154
4	8	3	0.45588	0.107958	−12.720715	9.561054	0.188162	−0.731982
5	12	6	−5.86847	0.107958	−12.853415	10	0.014074	−0.002154
6	−10	5	6.714694	0.107958	−12.864359	−9.610859	0.251364	2.879572
7	−10	6	−5.86368	0.107958	−12.865793	−9.612072	0.085958	2.875568

Table 3
The simulation results with white noise $N(0, 0.2)$ added to the penalized Shubert function evaluations

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	4.449063	0.107958	−3.739168	6.774669	2.23965	−12.185001
2	4	10	6.729869	0.107958	−12.79681	10	0.059367	−0.002154
3	8	5	0.427261	0.107958	−12.87015	10	0.023405	−0.002154
4	8	3	6.696855	0.107958	−12.84972	10	0.023574	−0.002154
5	12	6	−8.10426	0.107958	−3.749953	9.999982	0.021554	−0.002484
6	−10	5	6.704349	0.107958	−12.86835	−9.630264	0.261474	2.795851
7	−10	6	−5.84628	0.107958	−12.84872	−9.624987	0.234517	2.822746

$a = 2 \times 10^{-4}$. The proposed algorithm is used with $a = 0.01$ and the variance is updated according to the following equation.

$$\sigma_n = \frac{1}{\lfloor n/10 \rfloor^{1/3}},$$

Table 4
The simulation results with white noise $N(0, 0.3)$ added to the penalized Shubert function evaluations

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	5.46562	0.1079	−8.4960	6.6935	2.103081	−12.8358
2	4	10	6.70878	0.1079	−12.8701	10	0.037497	−0.00215
3	8	5	−8.1126	0.1079	−3.7457	10	0.04331	−0.00215
4	8	3	−5.8514	0.1079	−12.8638	10	0.017517	−0.00215
5	12	6	−7.1112	0.1079	−8.4467	10	0.021286	−0.00215
6	−10	5	6.72450	0.1079	−12.829	−9.6049	0.200885	2.898331
7	−10	6	−5.8533	0.1079	−12.8673	−9.6325	0.2594	2.783257

Table 5
The simulation results with white noise $N(0, 0.4)$ added to the penalized Shubert function evaluations

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	2.50023	0.107958	−2.639112	6.728462	2.145234	−12.806131
2	4	10	4.461055	0.107958	−3.74998	10	0.04247	−0.002154
3	8	5	6.699889	0.107958	−12.859456	9.999959	0.02963	−0.002901
4	8	3	6.726413	0.107958	−12.818587	10	0.01885	−0.002154
5	12	6	−5.842592	0.107958	−12.832664	10	0.02052	−0.002154
6	−10	5	−8.114598	0.107958	−3.743393	−9.615798	0.10445	2.862235
7	−10	6	−9.088435	0.107958	−2.728742	−9.622422	0.15321	2.834708

Table 6
The simulation results with white noise $N(0, 0.5)$ added to the penalized Shubert function evaluations

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	4.458778	0.107958	−3.749498	6.697289	2.232019	−12.851302
2	4	10	0.426808	0.107958	−12.870432	9.999987	0.02776	−0.002397
3	8	5	6.716193	0.107958	−12.860938	10	0.027162	−0.002154
4	8	3	6.693639	0.107958	−12.836154	9.692181	0.064142	−2.462594
5	12	6	0.42047	0.107958	−12.867394	10	0.015932	−0.002154
6	−10	5	−5.85382	0.107958	−12.868017	−9.604689	0.116198	2.897376
7	−10	6	6.710624	0.107958	−12.870028	−9.634266	0.259708	2.773433

where $\lfloor . \rfloor$ denotes the floor function. It can be easily verified that sequence $\{\sigma_n\}$ satisfies the conditions of Assumption 1. From the definition of the reinforcement signal β, it is clear that the highest expected reinforcement is 1.0 and the lowest expected reinforcement is 0.0 for either evaluation. It is easy that $M(\alpha) = E[\beta|\alpha]$ and

Table 7
The simulation results with white noise $N(0, 0.6)$

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	5.480349	0.089604	−8.517418	4.055911	20.009987	2.400759
2	4	10	0.435484	0.089604	−12.85371	4.054561	20.00997	2.415684
3	8	5	6.726655	0.089604	−12.81718	8.006934	20.009995	−0.62230
4	8	3	6.709202	0.089604	−12.87075	8.003345	20.009943	−0.70354
5	12	6	9.756602	0.089604	−2.727776	9.999975	20.009989	−0.00260
6	−10	5	−9.98241	0.089604	−2.205438	−9.93990	20.009995	−1.70987
7	−10	6	−7.09524	0.089604	−8.510625	−9.94036	20.009991	−1.71605

Table 8
The simulation results with white noise $N(0, 1.3)$

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.70339	0.089604	−12.8669	4.05304	20.009989	2.432262
2	4	10	0.43157	0.089604	−12.8642	4.05334	20.009985	2.428998
3	8	5	6.72627	0.089604	−12.8193	8.01151	20.009995	−0.51832
4	8	3	6.72909	0.089604	−12.8020	7.99121	20.009989	−0.97549
5	12	6	9.76507	0.089604	−2.72765	9.99998	20.009995	−0.00251
6	−10	5	−9.9842	0.089604	−2.22347	−9.9406	20.009995	−1.72048
7	−10	6	−9.9838	0.089604	−2.21967	−9.9455	20.009995	−1.78518

Table 9
The simulation results with white noise $N(0, 1.9)$

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.731426	0.089604	−12.78577	4.05399	20.009993	2.421903
2	4	10	6.685631	0.089604	−12.78776	4.05446	20.009977	2.416788
3	8	5	6.723327	0.089604	−12.83486	8.00724	20.009989	−0.61527
4	8	3	6.715631	0.089604	−12.86230	7.99826	20.009981	−0.81798
5	12	6	9.760613	0.089604	−2.728751	9.99999	20.009993	−0.00232
6	−10	5	−9.98213	0.089604	−2.202685	−9.9434	20.009998	−1.75658
7	−10	6	−9.98342	0.089604	−2.215217	−9.9444	20.009991	−1.77056

β satisfy Assumptions 2–4. The function evaluations are corrupted by a zero mean noise randomly generated from uniform and normal distributions. The results of simulation for two algorithms when noise in the range $[-0.5, 0.5]$ with uniform

Table 10
The simulation results with white noise $N(0, 2.6)$

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	5.47063	0.089604	−8.50962	4.054261	20.00998	2.418973
2	4	10	5.45497	0.089604	−8.44608	4.053005	20.00997	2.432638
3	8	5	6.70863	0.089604	−12.8708	8.005559	20.01000	−0.65347
4	8	3	6.72332	0.089604	−12.8348	8.004089	20.00998	−0.68673
5	12	6	9.77867	0.089604	−2.71016	9.999991	20.00999	−0.00231
6	−10	5	−9.9827	0.089604	−2.20913	−9.94208	20.01001	−1.73909
7	−10	6	−5.8538	0.089604	−12.8679	−9.94117	20.00996	−1.72688

Table 11
The simulation results with white noise $N(0, 3.2)$

Case	Initial values		The proposed algorithm			The old algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.703756	0.089604	−12.8675	4.056128	20.010014	2.398347
2	4	10	5.465978	0.089604	−8.49724	4.054752	20.01	2.413582
3	8	5	6.711122	0.089604	−12.8696	8.011147	20.010033	−0.52657
4	8	3	6.725053	0.089604	−12.8261	7.995277	20.009974	−0.88493
5	12	6	5.471429	0.089604	−8.51119	9.999896	20.010052	−0.00404
6	−10	5	−8.10843	0.089604	−3.74913	−9.93915	20.010021	−1.69975
7	−10	6	−9.98857	0.089604	−2.26330	−9.94576	20.009981	−1.78738

distribution are shown in Table 1, i.e. $f(x) = F(x) + U[-0.5, 0.5]$, where $U[a, b]$ is a random number in interval $[a, b]$ with uniform distribution. The results of simulation for two algorithms when noise with normal distribution are shown in Tables 2–11, i.e. $f(x) = F(x) + N(0, \sigma)$, where $N(0, \sigma)$ is a random number with uniform distribution with zero mean and the variance of σ. The simulation results show that the mean value of Gaussian distribution used in CALA always converge to a minimum of penalized Shubert function within the interval $[-10, 10]$ and has the higher rate of convergence than the algorithm given in [10].

By comparing the results of simulations for both algorithms, we conclude that two algorithms have approximately the same accuracy, but the proposed algorithm has higher speed of convergence. Figs. 2 and 3 show the changes in μ versus n for typical runs for the proposed algorithm and algorithm given in [10] when noise with uniform or normal distribution is added to the function evaluations. These figures show that the proposed algorithm converges to a minimum quickly and have oscillations around the minimum point. However, such oscillations could be decreased by using a small step size, which results in decreasing in the rate of convergence.

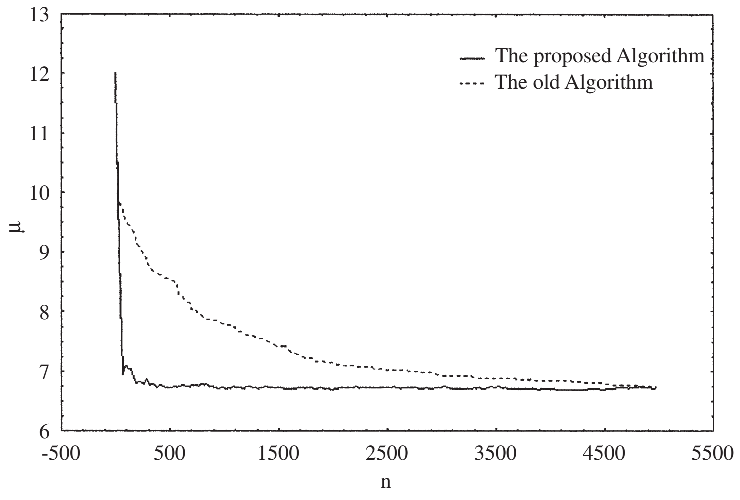

Fig. 2. The convergence of the mean value of the Gaussian distribution for uniform noise.

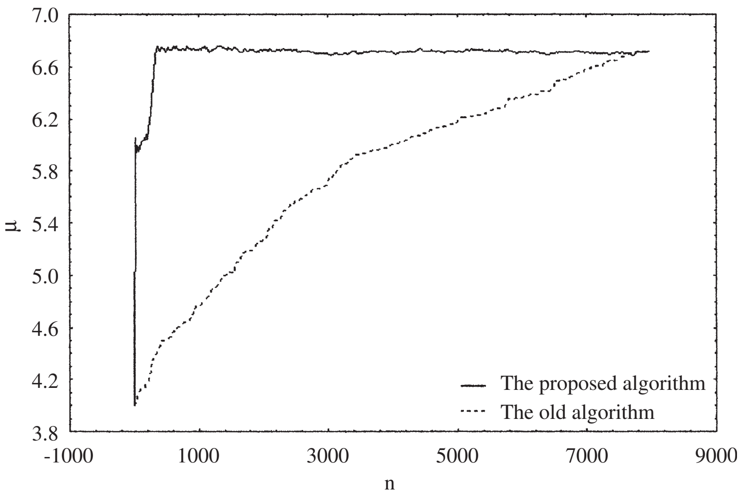

Fig. 3. The convergence of the mean value of the Gaussian distribution for noise with normal distribution.

5. Conclusions

In this paper, we have studied a random search method for solving stochastic optimization problems. We have considered a class of problems where the only available information is a noise-corrupted values of the function. In order to solve such problems, we proposed a new continuous action learning automaton and studied its convergence properties. We stated and proved a strong convergence theorem for the proposed learning automaton. Finally, we have given an algorithm for stochastic optimization and studied its behavior through computer simulations.

References

[1] H.J. Kushner, G.G. Yin, Stochastic Approximation Algorithms and Applications, Applications of Mathematics, Springer, New York, 1997.

[2] K.S. Narendra, K.S. Thathachar, Learning Automata: An Introduction, Prentice-Hall, New York, 1989.

[3] K. Najim, A.S. Pozyak, Learning Automata: Theory and Applications, Pergamon Press, Oxford, 1994.

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

[5] A.G. Barto, P. Anandan, Pattern-recognizing stochastic learning automata, IEEE Trans. Syst. Man Cybern. SMC-15 (1985) 360–375.

[6] M.A.L. Thathachar, P.S. Sastry, Learning optimal discriminant functions through a cooperative game of automata, IEEE Trans. Syst. Man Cybern. SMC-17 (1987) 73–85.

[7] K. Najim, A.S. Pozyak, Multimodal searching technique based on learning automata with continuous input and changing number of actions, IEEE Trans. Syst. Man Cybern.—Part B: Cybernetics 26 (1996) 666–673.

[8] V. Gullapalli, Reinforcement learning and its application on control, Ph.D. Thesis, Department of Computer and Information Sciences, University of Massachusetts, Amherst, MA, USA, February 1992.

[9] A. Vasilakos, N.H. Loukas, ANASA—a stochastic reinforcement algorithm for real-valued neural computation, IEEE Trans. Neural Networks 7 (1996) 830–842.

[10] G. Santharam, P.S. Sastry, M.A.L. Thathachar, Continuous action set learning automata for stochastic optimization, J. Franklin Inst. 331B (5) (1994) 607–628.

[11] K. Rajaraman, P.S. Sastry, Stochastic optimization over continuous and discrete variables with applications to concept learning under noise, IEEE Trans. Syst. Man Cybern.—Part A: Systems and Humans 29 (1999) 542–553.

[12] G.P. Frost, Stochastic optimization of vehicle suspension control systems via learning automata, Ph.D. Thesis, Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, Leicestershire, LE81 3TU, UI, October 1998.

[13] M.A.L. Thathachar, K.R. Ramakrishnan, A hierarchical system of learning automata, IEEE Trans. Syst. Man Cybern. SMC-11 (1981) 236–248.

[14] B.J. Oommen, E. Hansen, The asymptotic optimally of discretized linear reward-inaction learning automata, IEEE Trans. Syst. Man Cybern. SMC-14 (1984) 542–545.

[15] M.A.L. Thathachar, P.S. Sastry, A new approach to the design of reinforcement schemes for learning automata, IEEE Trans. Syst. Man Cybern. SMC-15 (1985) 168–175.

[16] G.I. Papadimitriou, A new approach to the design of reinforcement schemes for learning automata: stochastic estimator learning algorithm, IEEE Trans. Knowl. Data Eng. 6 (1994) 649–654.

[17] J.K. Lanctôt, B.J. Oommen, Discretized estimator learning automata, IEEE Trans. Syst. Man Cybern. 22 (1992) 1473–1483.

[18] B.J. Oommen, J.K. Lanctot, Discretized pursuit learning automata, IEEE Trans. Syst. Man Cybern. 20 (1990) 931–938.

[19] G.I. Papadimitriou, Hierarchical pursuit nonlinear automata with rapid convergence and high accuracy, IEEE Trans. Knowl. Data Eng. 6 (1994) 654–659.

[20] B.J. Oommen, M. Agache, Continuous and discretized pursuit learning schemes: various algorithms and their comparison, IEEE Trans. Syst. Man Cybern.—Part B: Cybernetics 31 (2001) 277–287.

[21] M. Agache, D.J. Oommen, Generalized pursuit learning schemes: new families of continuous and discretized learning automata, IEEE Trans. Syst. Man Cybern.—Part B: Cybernetics 32 (2002) 738–749.

[22] M.N. Howell, G.P. Frost, T.J. Gordon, Q.H. Wu, Continuous action reinforcement learning applied to vehicle suspension control, Mechatronics 7 (3) (1997) 263–276.

[23] O.V. Nedzelnitsky, K.S. Narendra, Nonstationary models of learning automata routing in data communication networks, IEEE Trans. Syst. Man Cybern. SMC-17 (1987) 1004–1015.

[24] M.S. Obaidat, G.I. Papadimitriou, A.S. Pomportsis, H.S. Laskaridis, Learning automata-based bus arbitration for shared-medium ATM switches, IEEE Trans. Syst. Man Cybern.—Part B: Cybernetics 32 (2002) 815–820.

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

[26] B.J. Oommen, E.V. de St. Croix, Graph partitioning using learning automata, IEEE Trans. Comput. 45 (1996) 195–208.

[27] M.R. Meybodi, H. Beigy, Solving stochastic shortest path problem using distributed learning automata, in: Proceedings of Sixth Annual Iran Computer Society of Iran Computer Conference CSICC-2001, Isfehan, Iran, February 2001, pp. 70–86.

[28] H. Beigy, M.R. Meybodi, Solving the graph isomorphism problem using learning automata, in: Proceedings of Fifth Annual International Computer Society of Iran Computer Conference, CISCC-2000, Tehran, Iran, January 2000, pp. 402–415.

[29] B.J. Oommen, T.D. Roberts, Continuous learning automata solutions to the capacity assignment problem, IEEE Trans. Comput. 49 (2000) 608–620.

[30] B.J. Oommen, T.D. Roberts, Discretized learning automata solutions to the capacity assignment problem for prioritized networks, IEEE Trans. Syst. Man Cybern.—Part B: Cybernetics 32 (2002) 821–831.

[31] M.R. Meybodi, H. Beigy, Neural network engineering using learning automata: determining of desired size of three layer feedforward neural networks, J. Faculty Eng. 34 (2001) 1–26.

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

[33] M.R. Meybodi, H. Beigy, New learning automata based algorithms for adaptation of back-propagation algorithm parameters, Int. J. Neural Syst. 12 (2002) 45–68.

[34] H. Beigy, M.R. Meybodi, Backpropagation algorithm adaptation parameters using learning automata, Int. J. Neural Syst. 11 (2001) 219–228.

[35] H. Beigy, M.R. Meybodi, Adaptive uniform fractional channel algorithms, Iran. J. Electrical Comput. Eng. 3 (2004) 47–53.

[36] H. Beigy, M.R. Meybodi, Call admission in cellular networks: a learning automata approach, Springer Lecture Notes in Computer Science, vol. 2510, Springer, Berlin, 2002, pp. 450–457.

[37] H. Beigy, M.R, Meybodi, A learning automata based dynamic guard channel scheme, Springer Lecture Notes in Computer Science, vol. 2510, Springer, Berlin, 2002, pp. 643–650.

[38] H. Beigy, M.R. Meybodi, An adaptive uniform fractional guard channel algorithm: a learning automata approach, Springer Lecture Notes in Computer Science, vol. 2690, Springer, Berlin, 2003, pp. 405–409.

[39] A. Papoulis, Probability, Random Variables, and Stochastic Processes, McGraw-Hill, New York, 1965.

[40] J.L. Doob, Stochastic Processes, Wiley, New York, 1953.