Genet Program Evolvable Mach
DOI 10.1007/s10710-013-9206-9

A new real-coded Bayesian optimization algorithm based on a team of learning automata for continuous optimization

Behnaz Moradabadi · Hamid Beigy

Received: 12 October 2012 / Revised: 24 August 2013

Abstract Estimation of distribution algorithms have evolved as a technique for estimating population distribution in evolutionary algorithms. They estimate the distribution of the candidate solutions and then sample the next generation from the estimated distribution. Bayesian optimization algorithm is an estimation of distribution algorithm, which uses a Bayesian network to estimate the distribution of candidate solutions and then generates the next generation by sampling from the constructed network. The experimental results show that the Bayesian optimization algorithms are capable of identifying correct linkage between the variables of optimization problems. Since the problem of finding the optimal Bayesian network belongs to the class of NP-hard problems, typically Bayesian optimization algorithms use greedy algorithms to build the Bayesian network. This paper proposes a new real-coded Bayesian optimization algorithm for solving continuous optimization problems that uses a team of learning automata to build the Bayesian network. This team of learning automata tries to learn the optimal Bayesian network structure during the execution of the algorithm. The use of learning automaton leads to an algorithm with lower computation time for building the Bayesian network. The experimental results reported here show the preference of the proposed algorithm on both uni-modal and multi-modal optimization problems.

Keywords Estimation of distribution algorithms · Bayesian optimization algorithm · Learning automata

B. Moradabadi (✉) · H. Beigy
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: behnazmoradabadi@gmail.com; moradabadi@ce.sharif.edu

H. Beigy
e-mail: beigy@sharif.edu

1 Introduction

In order to make a strongly search algorithm in evolutionary computing, a sort of evolutionary algorithms has been proposed. Genetic algorithm is a type of stochastic optimization techniques influenced by genetics and natural evolution [1]. In each generation of Genetic algorithm, a set of candidate solutions is selected and the new generation is produced using recombination (crossover) and mutation operators. Public, fixed, problem-independent mutation and recombination operators frequently lead to miss the knowledge of the relationship between variables, which is called building blocks, and result in convergence to local optima [2]. The building blocks are sub-problems of the optimization problem that show the relations and dependencies among the variables of the problem. In order to solve the above mentioned problem, the estimation of distribution algorithms (EDAs) are proposed [3]. These algorithms are stochastic optimization algorithms that construct an explicit probabilistic model from the candidate solutions found from the previous generations. Then, the new solutions are produced by sampling the constructed probabilistic model. Finally, the next generation will be produced from the sampled solutions using the replacement operator. This process is repeated until the stopping criteria have been reached. Several EDAs have been proposed for solving discrete and continuous problems [4, 5]. Extended compact genetic algorithm (ECGA) [6], factorized distribution algorithm (FDA) [7], and Bayesian optimization algorithm (BOA) [8] are examples of EDA algorithms proposed for discrete optimization problems. Estimation of Gaussian networks algorithm (EGNA) [9], mixed Bayesian optimization algorithm (MBOA) [10], (mixed) iterative density-estimation evolutionary algorithms (mIDEAs) [11], real-coded Bayesian optimization algorithm (rBOA) [12], and Bayesian networks and Gaussian mixture model (GMM) [13] are examples of EDA algorithms for continuous optimization problems.

In EGNA, a Gaussian network is constructed in each generation by the means of a scoring metric and the new generation is sampled from the constructed network [9]. The EGNA only uses one single-peak Gaussian model and hence is not appropriate for solving complex problems [9]. The MBOA tries to learn a Bayesian network from the selected individuals [10]. The structure of this network is a set of decision trees with univariate normal-kernel leaves. The MBOA generates one decision tree for each variable and then uses these decision trees to separate the problem space. The mIDEAs exploit Bayesian information criterion to select a probabilistic model and then employ a mixture of normal distributions to fit the model [11]. The mIDEAs use a mixture of normal distributions by clustering the candidate solutions and hence cannot preserve the population building block crossover (PBBC). PBBC is a kind of population-wise building-block crossover that uses a probability distribution calculated from the building-blocks [3, 4]. In the sampling phase, mIDEAs select a distribution (a cluster) of a mixture of normal distributions and generate the new individuals from it. Hence, these algorithms cannot realize the PBBC unless one cluster contains most of building blocks. In rBOA, the promising solutions are selected from the current population using a parent selection method, and a Bayesian network is constructed to model the distribution of the promising solutions [12]. Then the new individuals are sampled from the constructed Bayesian network using the following procedure. In the sampling phase, the sub-trees (building blocks of the problem) of the Bayesian network are extracted and the whole solutions of each sub-tree are clustered to make a mixture of normal distributions. The new partial solutions of each sub-tree are sampled from the corresponding mixture of normal distributions. Finally, the new individuals are merged into the main population and replaced some of the old ones. The above steps are repeated until some stopping criteria are satisfied [12]. Unlike mIDEAs, which consider a mixture of normal distributions of the population in the model sampling phase, rBOA uses a mixture of normal distributions of each building block in the model sampling phase. So, rBOA can better preserve the PBBC in comparison to the mIDEAs in continuous optimization problems.

Bayesian networks and Gaussian mixture model (GMM) is an evolutionary continuous optimization algorithm that uses a Bayesian network to model the dependencies between the variables of the problem. Then, one GMM is adopted to model the probability distribution of each sub-problem and the new generation is produced from the GMM of each sub-problem [13].

In [14, 15], an improved version of real-coded Bayesian optimization algorithm, called IrBOA, has been proposed. The IrBOA utilizes an adaptive clustering method to decompose the problem space into multiple Bayesian networks. Each Bayesian network has its own structure and parameters. The adaptive clustering method results in searching the problem space efficiently in the starting generations of the algorithm and also improves the quality of solutions with realizing the PBBC in the next generations of the algorithm. The use of adaptive clustering leads to realizing the PBBC; because when the diversity of solutions is decreased, the number of clusters is also decreased. Hence, a fewer number of Bayesian networks is generated and more building blocks are aggregated in one Bayesian network [14].

In many EDAs, generating the model of the dependencies and independences of the variables of a problem is considered as a bottleneck; because in many situations, finding the best model corresponding to the candidate solutions is a difficult combinatorial problem. Usually, many EDAs use greedy algorithms to create the model. So, using some techniques for generating a model is a valuable context in EDAs.

On the other hand, learning automaton (LA) is an adaptive decision making unit that tries to learn the optimal action from a set of allowable actions by interacting with a random environment [16]. In each step, it selects an action from its action-set. The action selection in the LA is based on a probability distribution over the action-set. The selected action is applied to the environment and then a reinforcement signal is produced by the environment. LA updates the probability distribution of its actions according to both reinforcement signal and a learning algorithm and again chooses an action. These steps are repeated until LA converges to some action.

In this paper, we propose a new real-coded Bayesian optimization algorithm based on the learning automaton, called LA-rBOA, to optimize the real-valued functions in continuous domains. The proposed algorithm uses a team of LAs to build the Bayesian network. For a successful application, the rBOA must learn a network that reflects the linkage of the problem properly. The problem of learning the structure of a network based on a scoring metric is a difficult combinatorial problem [17]. So, there is no known polynomial-time algorithm to determine the optimal structure of a Bayesian network with respect to the most scoring metrics and usually a simple greedy algorithm is used to generate the Bayesian network [18]. This paper utilizes a team of LAs to produce the Bayesian network instead of the greedy algorithms. It uses one LA for each possible edge in the Bayesian network. Each LA determines either the corresponding edge should be appeared in the Bayesian network or not. The Bayesian network is built according to the selected actions of learning automata. Using LA for generating Bayesian network leads to generating Bayesian network in lower computation time. Then the new individuals are sampled from the constructed Bayesian network. Finally, the probability distribution of actions of each LA is updated according to the impact of the corresponding edge in the produced Bayesian network. Because of the nature of LA, it tries to learn the optimal behavior and so the optimal Bayesian network structure. These steps are repeated until the action of each LA converges to some value. The experimental results show that the LA-rBOA is superior to the some EDAs such as IrBOA and mIDEA and some other population based algorithms [13, 19, 20] in term of accuracy and performance.

The rest of the paper is organized in the following way. Section 2 explains the Bayesian Networks and LA is described briefly in Sect. 3. Section 4 presents the proposed real-coded Bayesian optimization algorithm based on a team of LAs and Sect. 5 reports the experimental results obtained with the LA-rBOA. The paper concludes with a conclusion given in Sect. 6.

2 Bayesian networks

A Bayesian optimization algorithm constructs a Bayesian network using a set of promising solutions and then samples the next generation from the constructed Bayesian network [21]. Various algorithms can be used to build the Bayesian network. Also there is a variety of methods can be used to measure the quality of the constructed network. Many of these metrics can use the prior information about the problem to improve the quality of the network and the efficiency of the algorithm [21].

A Bayesian network $M = (\zeta, \theta)$ consists of a structure ζ and a set of parameters θ. The structure of a network is described by a directed acyclic graph with a set of nodes corresponding to the problem variables and a set of edges corresponding to the conditional dependencies between variables. The parameters of the network (θ) is a set of probability distributions that is described as a product of probability density functions. Learning the structure of Bayesian networks has two components [5]: a scoring metric and a search procedure. Scoring metric evaluates the quality of the constructed Bayesian network. In other words, it determines how much the structure of the constructed Bayesian network fits the promising solutions. Search procedure determines an algorithm to find a network that better fits the promising solutions with respect to a given scoring metric. In the rest of this section, we briefly describe each of these two components.

2.1 Scoring metric

There is a variety of metrics proposed for evaluation of Bayesian networks such as Bayesian metrics, Minimum description length metrics, and Bayesian information criterion, to mention a few. Bayesian metrics measure the quality of the Bayesian network by computing a marginal likelihood of the Bayesian network with respect to the given data and inherent uncertainties [5, 12, 21]. Minimum description length metrics are based on the assumption that the number of regularities in the data encoded by a model is somehow proportional to the amount of data compression allowed by the model [21]. The Bayesian information criterion (BIC) is a criterion for model selection among a finite set of models and it is based on the likelihood function.

Let's consider a d-dimensional real-value optimization problem. The joint probability density function (pdf) of this optimization problem can be written according to Eq. (1):

$$f_{M(\zeta,\theta)}(Y) = \prod_{i=1}^{d} f_{\theta^i}(Y_i|\Pi_i), \tag{1}$$

where $Y = (Y_1, \ldots, Y_d)$ represents a vector of real-valued random variables, Π_i is the set of parents of node $\mathrm{Y_i}$, the set of nodes from which there exists an edge to Y_i, and $f_{\theta^i}(Y_i|\Pi_i)$ is the conditional pdf of Y_i given Π_i with parameters θ_i [5, 12, 21]. Let $S = \left(Y^0, Y^1, \ldots, Y^{|S|-1}\right)$ be the set of promising solutions, the BIC metric is given by the following equation:

$$BIC\left(f_{M(\zeta,\theta)}, S\right) = -\ln\left(\prod_{j=0}^{|S|-1} f_{M(\zeta,\theta)}\left(\mathrm{Y^j}\right)\right) + \gamma \ln(|S|)|\theta|, \tag{2}$$

where γ regularizes the extent of penalty [12]. This metric is a combination of the model fitting error and the model complexity and must be minimized. Since the minimal negative log-likelihood is equivalent to minimal entropy, Eq. (2) can be rewritten as the following equation

$$BIC\left(f_{M(\zeta,\theta)}, S\right) = |S| H\left(f_{M(\zeta,\theta)}(Y)\right) + \gamma \ln(|S|)|\theta|, \tag{3}$$

where $\mathrm{H}(\mathrm{f}_{\mathrm{M}(\zeta,\theta)}, (\mathrm{Y}))$ represents the differential entropy of $\mathrm{f}_{\mathrm{M}(\zeta,\theta)}(\mathrm{Y})$ [12, 22, 23]. In order to search the problem space efficiently, a (joint) mixture distribution is used for modeling the promising solutions. Hence, Eq. (3) can be rewritten as:

$$BIC\left(f_{M(\zeta,\theta)}, S\right) = \sum_{j=0}^{k} |S_j| H\left(f_{M\left(\zeta,\theta_j\right)}(Y)\right) + k\gamma \ln(|S|)|\theta_j|, \tag{4}$$

where k is the number of mixture components, $|\mathrm{S_j}|$ is the expected number of the promising solutions drawn from a probability distribution $\mathrm{H}\left(\mathrm{f}_{\mathrm{M}\left(\zeta,\theta_j\right)}(\mathrm{Y})\right)$, and $|\theta_\mathrm{j}|$ is the size of the parameters of jth mixture component [22, 23].

2.2 Search procedure

Given a scoring metric, the problem of learning the structure of a Bayesian network belongs to the class of NP-hard problems and there is no known polynomial-time algorithm for finding the best network structure corresponding to most scoring metrics [17]. Usually, a simple, greedy algorithm is used to build the network [18, 23]. The greedy algorithm adds an edge with the greatest improvement of the current network quality in each step until no more improvement is possible. The initial network structure can be a graph with no edges; also, it can take advantage of using the prior information such as using the best tree computed by the polynomial-time maximum branching algorithm [5, 18]. Since Bayesian networks are acyclic graphs, after each operation, the graph structure must be validated and all cycles must be removed from the constructed graph [12, 24, 25].

3 Learning automata

Learning automaton is an adaptive decision making device that tries to learn the optimal action from a set of allowable actions in an unknown random environment [16]. This process is done by interacting between an automaton and a random environment. At each time t, the automaton selects an action, $\alpha(t)$ using its action probability distribution, $p(t)$, and applies the selected action to the random environment. Then, the random environment generates a stochastic reinforcement signal, $\beta(t)$, to the automaton [14]. The automaton updates the action probability distribution using both the reinforcement signal and a learning algorithm. Figure 1 shows the interaction between a learning automaton and its environment. Learning automaton based on to its action-set can be classified into two major classes: finite action-set LA (FALA) and continuous action-set LA (CALA) [26–28]. FALA has finite action-set but CALA has actions chosen from real line. For an r-action FALA, the action probability distribution is represented by an r-dimensional probability vector and is updated by the learning algorithm. For a CALA, the action probability distribution is represented by a continuous function and this function is updated by the learning algorithm at any time. Learning automata have been used successfully in many applications such as routing and call admission control in computer network [29, 30], solving NP-Complete problems [31–33], and too many other applications [26, 34–39]. For further information about LA please refer to [39].

4 The proposed real-coded Bayesian optimization algorithm

The problem of finding the optimal Bayesian network corresponding to the variables dependencies of a problem belongs to the class of NP-hard problems, and Bayesian

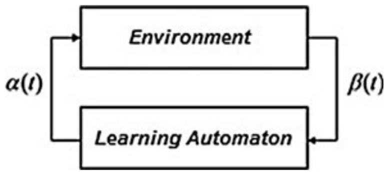

Fig. 1 The interaction between learning automaton and the environment

optimization algorithms typically use greedy algorithms to generate Bayesian networks; these greedy algorithms take long execution time. In this section, we propose a new real-coded Bayesian optimization algorithm (LA-rBOA), which is based on a team of learning automata for building the Bayesian network. The proposed algorithm uses a team of learning automata to choose the structure of the Bayesian network instead of using a greedy algorithm. This algorithm leads to decreasing the execution time for constructing Bayesian network. The proposed algorithm has two main phases: model building and sampling phases as described below. In model building phase, the team of learning automata chooses a structure and then based on the effectiveness of each edge a separate reinforcement signal is produced. Finally the team of learning automata updates the action probability vectors of all learning automata according to the received signal and a learning algorithm. The effectiveness of the each edge is estimated by using the promising solutions of the current generation. After building the structure of the network, in sampling phase the offspring are generated by sampling from the constructed network. Finally, the new population is selected according to the fitness evaluation. The pseudo code of the proposed algorithm is given in Algorithm 1. In the following sections, we describe the model building and the model sampling phases in detail.

Algorithm 1: Pseudo code of the proposed algorithm (LA-rBOA)

```
1: Let $d$ be dimension of the problem, Np be population size, and $m$ be the number of LAs, i.e. $m = d \times (d - 1)$;
2: Let $t$ be the generation counter and initially set to 0 and $Max_{NFC}$ be the total number of generations.
3: Create a random population of $Np$ individuals.
4: Set the initial actions probability distribution of each learning automaton to be uniform.
5: while $t < Max_{NFC}$ do
6:     Generate LAs' action, $(\alpha_1(t), \alpha_2(t), \ldots, \alpha_m(t))$.
7:     Build the structure of the Bayesian network based on LAs' actions and remove all cycles from the constructed network
8:     Select $\tau$ % individuals from population based on their fitness.
9:     Update the actions probability distribution of each learning automaton $A_j$ based on the reinforcement signal $(\beta(j, t))$.
10:    Extract all trees (sub-problems) in the constructed Bayesian network.
11:    Cluster the partial solutions corresponding to each tree (sub-problem) and assign a score v(i,r) to cluster $C_i$ of tree r.
12:    for $n=1$ to $Np$ do
13:        for $r=1$ to number of sub-problems do
14:            Select a cluster $C_i$ of sub-problem r based on the coefficient score v(i,r)
15:            Generate the partial solution of nth individual based on the conditional normal distribution of cluster $C_i$.
16:        end for
17:        Merge all partial solutions together to create a new individual.
18:    end for
19:    Merge the new and the old generations, evaluate its fitness, and choose the best $Np$ individuals as the new generation.
20:    Set $t = t + 1$
21: end while
```

4.1 Model building

This step tries to find a model, $M = (\zeta, \theta)$, that better fits the promising solutions. Finding the structure of the network has two components: a scoring metric and a search procedure [5, 18, 21, 23]. The proposed algorithm (LA-rBOA) uses the BIC score given in Eq. (4) for the scoring metric and a team of learning automata to generate the Bayesian network structure as a search procedure. In what follows, we describe the search procedure.

Let d be the dimension of the real-valued optimization problem. Hence, the corresponding Bayesian network has d nodes. Since the Bayesian network is a directed graph, the maximum number of edges of the network with d nodes is $m = d \times (d - 1)$. We use a team of m learning automata $(\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_\mathrm{m})$ with action-set {0,1} and each learning automaton is associated to one edge. Let the action of automaton A_j at time t denoted by $\alpha_j(t)$ and it determines either the corresponding edge should be appeared in the Bayesian network or not. Action $\alpha_j(t) = 0$ means that the corresponding edge does not appear in the Bayesian network while value $\alpha_j(t) = 1$ means that the corresponding edge appears in the network. The action probability vector of automaton $A_j(for\, j = 1, 2, \ldots, m)$ at time t is denoted by $p^j(t) = \left(p_0^j(t), p_1^j(t)\right)$, where $p_i^j(t)$ is the probability of choosing action α_i (for $i = 0,\ 1$) at time t. At the first iteration, because there is no prior information about the problem variables dependencies, we use the uniform probability distribution to select actions; i.e. $p_i^j(0) = 0.5$ (for $j = 1,\ 2,\ \ldots,\ m$ and $i = 0,\ 1$). The initial action probability vector for each automaton A_j can also be configured based on the prior information about the variables dependencies of the problem. After selecting actions for all learning automata, we use this information to build a Bayesian network structure. The search procedure uses the selected action of each automaton to determine either the corresponding edge must appear in the Bayesian network or not. Now, the constructed Bayesian network structure may have cycles. In order to remove cycles, the search procedure uses depth first search algorithm (DFS) with a random start node to navigate the current structure and mark all of the back edges which point from a node to one of its ancestors After detecting all back edges, the search procedure removes all. Because of choosing a random node in the depth search algorithm, the proposed algorithm has the chance of removing different edges in the execution of the algorithm to choose the structure of the network without biasing to the any edge in the network.

In the next step, the reinforcement signal for A_j is calculated based on the estimated influence of the corresponding edge on the score of the constructed Bayesian network. In order to do this, τ % of the best current solutions are selected and a reinforcement signal is calculated by the difference between the BIC score of the current Bayesian network and the Bayesian network that is constructed by removing the edge corresponding to A_j. The estimated influence of the A_j at time t, $R(j,\ t)$, is calculated based on the selected promising solutions using the following equation:

$$R(j,t) = BIC(K, f_{M(\zeta,\theta)}(Y), S) - BIC(K, f_{M(\zeta,\theta)_{-j}}(Y), S), \tag{5}$$

where $BIC(K, f_{M(\zeta,\theta)}(Y), S)$ is the BIC score of the constructed Bayesian network and $BIC(K, f_{M(\zeta,\theta)_{-j}}(Y), S)$ is the BIC score of the Bayesian network that is created by removing the edge corresponding to automaton A_j. Hence, the reinforcement signal of the A_j at time t, (j, t), is the normalized value of $R(j, t)$ calculated using the following equation:

$$\beta(j,t) = Max\left(0, \frac{R(j,t) - Min(j)}{Max(j) - Min(j)}\right), \tag{6}$$

where Min(j) and Max(j) are the minimum and maximum of the estimated influence of A_j up to time t. According to the given reinforcement signal, learning automata with larger reinforcement signal value have more impact on the Bayesian network structure. Then the algorithm updates the action probability distribution of each automaton using the following equation:

$$p_i^j(t+1) = \begin{cases} [(p_i^j(t) + (1-\lambda)\beta(j,t))]\omega(t) & \alpha_j(t) = i \\ 1 - p_i^j(t) & \alpha_j(t) \neq i \end{cases} \tag{7}$$

where λ is a constant parameter and $\omega(t)$ is parameter to guarantee that $p^j(t+1)(for\, j = 1, 2, \ldots, m)$ remains a probability vector.

4.2 Model sampling

After construction of a Bayesian network, new generation is sampled from the Bayesian network. The sampling method of the proposed algorithm is the same as the method used in the improved real-coded BOA algorithm (IrBOA) [14, 15]. In order to do this, first all maximal connected sub-graphs are extracted from the constructed Bayesian network. These sub-graphs determine the sub-problems whose variables have interactions. In order to simplify the presentation, the promising solutions of a sub-problem are called the partial solutions of that sub-problem. Then the partial solutions of each sub-problem are partitioned into some groups using a clustering algorithm. The clustering the partial solutions of each sub-problem results in realizing the non-linearity and searching the space effectively [12, 21]; because the non-linear dependencies can be modeled with a combination of piecewise linear interaction models. Also, the clustering has advantage of searching the problem space efficiently [14, 15]. It considers the partial solutions of each sub-problem independently and so it can implement the population building block crossover (PBBC). For clustering the partial solutions, the algorithm utilizes an adaptive clustering method, k'-means algorithm [40]. k'-means clustering is an extended version of k-means clustering in which there is no need to determine the number of clusters as a parameter because the correct number of clusters is determined during the execution of the clustering algorithm. k'-means algorithm uses the density of clusters to determine the correct number of clusters. If the ith cluster, denoted by C_i, has n_i partial solutions in a sub-problem with N partial solutions then the amount information of this cluster denoted by $I(C_i)$, is calculated using the following equation:

$$I(C_i) = |\log(ni/N)|. \tag{8}$$

This information is used to approximate uncertainty about C_i [40]. The k'-means algorithm has two phases: in the first phase, k'-means uses k-means algorithm to determine the initial cluster centers. The number of initial clusters in k. -means is determined with parameter k' that is greater than the correct number of clusters [40]. In the second phase, for each partial solution xt and all k'. clusters, k'-means calculates the cluster membership function $I(xt, j)$ using Eq. (9):

$$I(x_t,j) = \begin{cases} 1 & if\, j = \underset{i}{argmin}\{d(x_t, C_i)\}\; i = 1, 2, \ldots, k\prime \\ 0 & otherwise \end{cases}, \tag{9}$$

where $d(x_t, C_i)$ is a distance metric and calculated using the following equation:

$$d(x_t, C_i) = x_t - C_i - Elog(I(C_i)), \tag{10}$$

and E is a constant [40]. Then every paral solution of a sub-problem is grouped into the cluster whose centroid is closest to it based on the cster membership $I(xt, j)$. So, the center of cluster C_i is calculated as the mean value of its members as given below:

$$C_i = \frac{1}{|C_i|} \sum_{x_t \in C_i} x_t \quad for\left(i = 1, 2, \ldots, k^{'}\right), \tag{11}$$

where k' is the current number of clusters. These two phases are repeated until all cluster centers remain fixed for all of the partial solutions [40].

After clustering the partial solutions of each sub-problem, a coefficient score is assigned to each cluster according to the ratio of the number of partial solutions of a cluster in sub-problem r to the whole partial solutions of sub-problem r. This score is used as the probability of choosing a cluster to generate partial solutions. In other words, if the partial solutions of sub-problem r are grouped in k' clusters and cluster C_i has $n_{r,i}$ partial solutions then coefficient score of cluster C_i corresponding to the sub-problem r denoted by $v(r, i)$ and written by the following equation.

$$v(r, i) = \frac{n_{r,i}}{\prod_{i=1}^{k'} n_{r,i}}. \tag{12}$$

In order to generate a partial solution corresponding to sub-problem r, the LA-rBOA chooses cluster C_i with probability $v(r, i)$. It first employs the normal probability distribution function to the partial solutions of the selected cluster and then generates the new partial solutions from the distribution [12, 21, 23]:

$$f(Y_0|Y_1, Y_2, \ldots, Y_n) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(Y_0-\mu_0)^2}{2\sigma^2}} \tag{13}$$

where Y_0 is the partial solution that must be produced, Y_1, Y_2, …, Y_n are the parents of Y_0,

$$\sigma = \frac{1}{\sqrt{(\sum^{-1})_{0,0}}}, \tag{14}$$

$$\mu = \mu_0 - \frac{\sum_{j=1}^{Y_n} \left(Y_j - \mu_j\right)_{j,0}}{(\sum^{-1})_{0,0}} \tag{15}$$

and μ_j is the mean of Y_j and is the inverse of covariance matrix for $(Y_0|Y_1, Y_2, \ldots, Y_n)$ of cluster C_i.

5 Experimental results

In order to evaluate the performance of the proposed algorithm, computer experiments have been conducted and the performance of the proposed algorithm has been compared with the performance of some related algorithms on the standard benchmark test functions (CEC2005 [41] and CEC2011 [42] test functions). In these experiments, we use the quality of solutions, the computation time, and the convergence rate as the performance criteria. In the rest of this section, we first give the test functions and then give a set of six experiments. In the first experiment, we study the sensitivity of the proposed algorithm on its parameters. In the second experiment, the performance of the proposed algorithm compared with the performance of some related EDAs on the CEC2005 test functions. In the third experiment, the performance of the LA-rBOA is compared with the performance of some population based algorithms on the CEC2005 test functions. In the fourth and fifth experiments, the computation time and convergence rate of the LA-rBOA is compared with some EDAs on the CEC2005 test functions, respectively. Finally, in the sixth experiment, the performance of the LA-rBOA is compared with some algorithms used in IEEE CEC2011 competition on the CEC2011 test functions.

5.1 Test functions

In order to compare the performance of algorithms CEC2005 and CEC2011 test functions have been used. In what follows, these test functions are given briefly.

5.1.1 CEC2005 test functions

This set of test functions is chosen from the benchmark functions used in CEC2005 [41]. This set includes five uni-modal (F1–F5) and twenty multi-modal (F6–F25) functions and the quality of solution is used as the performance measure. Each function in the test suite presents a real-value function that must be minimized and the global optimum of all functions is at the origin. The information of this test suit is given in Table 1. This benchmark has two important parameters: the dimension of each function denoted by *d* and the shift value of each function denoted by FBias. In the experiments given in the paper, *d* and FBias are set to 30 and 0, respectively.

5.1.2 CEC2011 test functions

The CEC2011 benchmark has 22 real-world optimization functions and used to evaluate algorithms submitted in IEEE CEC2011 [42]. These problems include a parameter estimation problem for frequency-modulated sound saves (T1); a Lennard–Jones potential problem (T2); a bifunctional catalyst blend control problem (T3); a stirred tank reactor control problem (T4); two Tersoff potential

Table 1 The CEC2005 benchmark functions

Function	Function name	Range
F1	Shifted sphere function	[−100, 100]
F2	Shifted Schwefel's problem	[−100, 100]
F3	Shifted rotated high conditioned elliptic function	[−100, 100]
F4	Shifted Schwefels problem with noise in fitness	[−100, 100]
F5	Schwefels problem with global optimum on bounds	[−100, 100]
F6	Shifted Rosenbrocks function	[−100, 100]
F7	Shifted rotated Griewanks function without bounds	[0, 600]
F8	Shifted rotated Ackleys function with global optimum on bounds	[−32, 32]
F9	Shifted Rastrigins function	[−5, 5]
F10	Shifted rotated Rastrigins function	[−5, 5]
F11	Shifted rotated Weierstrass function	[−0.5, 0.5]
F12	Schwefels problem 2.13	[−π, π]
F13	Expanded extended Griewanks plus Rosenbrocks function (F8F2)	[−3, 1]
F14	Shifted rotated expanded Scaffers F6	[−100, 100]
F15	Hybrid composition function	[−5, 5]
F16	Rotated version of hybrid composition function	[−5, 5]
F17	Rotated hybrid composition function with noise in fitness	[−5, 5]
F18	Rotated hybrid composition function	[−5, 5]
F19	Rotated hybrid composition function with a narrow basin for the global optimum	[−5, 5]
F20	Rotated hybrid composition function with the global optimum on the bounds	[−5, 5]
F21	Rotated hybrid composition function	[−5, 5]
F22	Rotated hybrid composition function with high condition number matrix	[−5, 5]
F23	Non-continuous rotated hybrid composition function	[−5, 5]
F24	Rotated hybrid composition function	[−5, 5]
F25	Rotated hybrid composition function without bounds	[2, 5]

minimization problems (T5 and T6); a radar poly-phase code design problem (T7); a transmission network expansion problem (T8); a transmission pricing problem (T9); an antenna array design problem (T10); two dynamic economic dispatch problems (T11.1 and T11.2); five static economic dispatch problems (T11.3–T11.7); three hydrothermal scheduling problems (T11.8–T11.10); and two spacecraft trajectory optimization problems (T12 and T13). For additional information about this set of benchmark functions, please refer to the Ref. [42].

5.2 Sensitive analysis

In this section, we study the effect of each parameter on the performance of the LA-rBOA. The proposed algorithm has six parameters: (1) Selection strategy method, (2) Parameter τ in the selection strategy, (3) Population size (Np), (4) Complexity factor in the BIC score (γ), (5) The number of Mixture components (k) and (6) The decay Factor in the learning rule of learning automata (λ). In order to study the effect of one parameter, the values of other parameters are set fixed. The default values of the parameters are listed in the following setting: (1) parent selection method is truncation selection (2) parameter $\tau = 20$, (3) population size (Np) = 2,000, (4) complexity factor of BIC score $\gamma = 0.5$, (5) the number of Mixture components (k) = 30, (6) decay factor of all learning automata be $\lambda = 0.4$. In what follows, we study the effect of each parameter on the performance of the algorithm.

The effect of different selection strategies: The goal of this experiment is to study the performance of the LA-rBOA based on different selection strategies. Comparison is based on truncation and tournament selection methods. In tournament selection, τ individuals are randomly picked from the population and the best one is selected as a candidate solution. In truncation selection the best τ % individuals of the population are selected as the candidate solutions. Figure 2a shows the number of function evaluations needed to optimize function F9 with respect to different selection strategies. Based on the results, LA-rBOA has better performance on the truncation selection method than the tournament selection. This parameter also has the same effect on other test functions.

The effect of parameter τ in the truncation selection: This experiment compares the impact of parameter τ in truncation selection on the performance of the LA-rBOA. To do this, parameter τ is varied from the set {20 %, 40 %, 60 %}. and the maximum number of function calls is calculated for function F9. Figure 2b presents that the best value of parameter τ for function F9 is 20. The same behavior is also observed in other functions.

The effect of population size (Np): This experiment compares the impact of parameter Np on the performance of the LA-rBOA. To do this, value of F9 is chosen from the set $\{500, 1000, 1500, \ldots, 5000\}$ and the results for function F9 are presented. Figure 2c presents the best value of parameter Np for function F9 is between 2,000 and 5,000. This parameter also has the similar effect on other test functions.

The effect of complexity factor in the BIC score (γ): This experiment compares the effect of parameter γ on the performance of the LA-rBOA on function F9. Figure 2d shows that the best value of the parameter γ for function F9 is between 0.4 and 0.5. The same behavior was observed on other functions.

The effect of the number of mixture components (K): The influence of the number of mixture components on the performance of the LA-rBOA is studied for function F9 and the result is shown in Fig. 2e. This figure represents that the best value of

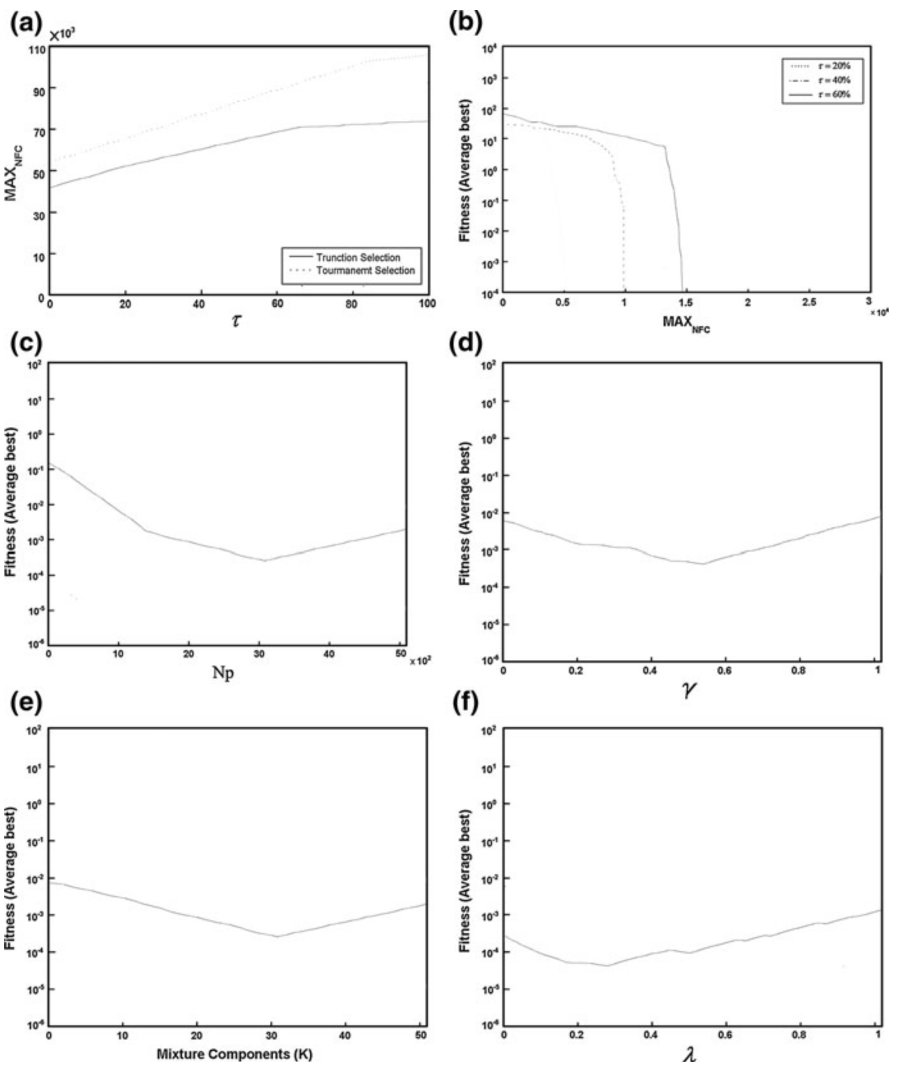

Fig. 2 Parameters analysis in the LA-rBOA. **a** Different selection strategies on function F9. **b** Parameter τ on function F9. **c** Population size, Np, on function F9. **d** Parameter γ on function F9. **e** Mixture components, K, on function F9. **f** Parameter λ on function F9

parameter k, for function F9 is about 30. The same behavior was also seen on other test functions.

The effect of decay factor in the LA learning rule (λ): Finally, the effect of parameter λ on the performance of the LA-rBOA on function F9 is presented in Fig. 2f and it shows that the best value of parameter λ for function F9 is between 0.2 and 0.4. This parameter also has the same effect on other test functions.

5.3 Comparison of the LA-rBOA with some EDA algorithms

In this section, LA-rBOA is compared with the improved real-coded BOA (IrBOA) [14] [15] and the mIDEA [11] on the CEC2005 test functions. Based on No Free Lunch theorem: "for any algorithm any elevated performance over one class of problems is offset by performance over another class" [43]. So, the LA-rBOA tries to solve a class of benchmark functions in term of accuracy and efficiency. For all conducted experiments, the parameters of the LA-rBOA are chosen empirically based on the sensitivity analysis given in the previous section. Parameter setting for the conducted experiments are: (1) parent selection method is truncation selection (2) parameter $\tau = 20$, (3) population size (Np) = 2,000, (4) complexity factor in BIC score $\gamma = 0.5$, (5) the number of Mixture components (k) = 30, (6) decay factor in LA learning rule $\lambda = 0.4$. Also, two points should be mentioned: (1) the initial population in all algorithms is created based on the uniform distribution on the constraints of each function, and (2) in order to have a fair comparison, the maximum number of function calls, $\mathrm{Max}_{\mathrm{NFC}}$, is set to $1{,}500 \times d$.

In this experiment, the parameters setting of the IrBOA and the mIDEA are borrowed from [12, 15]. The results for the LA-rBOA, the IrBOA and the mIDEA on the uni-modal functions F1–F5 are summarized in Table 2. For each function the mean error value and the standard deviation of obtained fitness values on the 50 independent runs are presented. It should be mentioned that for the following tables given in this section, the best results are highlighted. Also, in order to have a better comparison, a set of two-tailed *t* test were taken; because the *t* test determines the difference between the means of two independent samples. The *t* test can be used to check whether the two algorithm means are different and calculated using the following equation:

$$\text{t - test}(\mathrm{A1}, \mathrm{A2}) = \frac{\hat{\mathrm{X}}_1 - \hat{\mathrm{X}}_2}{\sqrt{\frac{\mathrm{s}_1^2}{\mathrm{n}_1} + \frac{\mathrm{s}_2^2}{\mathrm{n}_2}}}, \tag{16}$$

where $\hat{X}_i, s_i^2$ and $\mathrm{n_i}$ are the means, standard deviation, and the number of runs of algorithm *i*, respectively. In this experiment, we consider the LA-rBOA as algorithm *A*1. Hence, if the result of *t* test between the LA-rBOA and another algorithm is a negative value then the LA-rBOA statistically outperforms another algorithm. The statistical differences between the experiments obtained with the LA-rBOA, the IrBOA, and the mIDEA for functions F1–F5 are listed in the Table 3. Tables 2 and 3 show that in most uni-modal functions, the LA-rBOA clearly outperforms the IrBOA and the mIDEA.

Function F1 is shifted Sphere function, function F2 is shifted Schwefel's problem 1.2, and the function F3 is shifted rotated high conditioned elliptic function. These three functions have different conditions which make F3 to be harder than F2 and F2 to be harder than F1. From the results, we could conclude that LA-rBOA outperforms on the two other algorithms for functions F1 and F2. Also, in function F3, a better result is achieved for the LA-rBOA. Function F4 is shifted Schwefel's problem 1.2 with noise in fitness. Noise in the fitness disturbs the search process. From the results, we observe that the mIDEA and IrBOA are

Table 2 Numerical results for the LA-rBOA, the IrBOA and the mIDEA

Function	LA-Rboa		IrBOA		mIDEA	
	Mean	Std	Mean	Std	Mean	Std
F1	**3.70e−12**	2.50e−11	9.53e−10	4.63e−10	9.47e−9	4.45e−9
F2	**6.34e−11**	5.74e−11	6.94e−10	1.49e−10	9.85e−9	1.53e−10
F3	**2.96e−30**	2.82e−36	11.0e−30	42.01e−23	6.87e−25	1.40e−19
F4	5.52e+0	0.80e+0	**8.13e−7**	1.74e−6	1.11e−3	4.35e−5
F5	**5.71e−8**	5.42e−11	13.00e−2	0.05e+0	1.40e−1	4.35e−2

Table 3 The results of statistical test for the LA-rBOA, the IrBOA and the mIDEA

Function	IrBOA	mIDEA
F1	**−1.10e+11**	**−1.20e+10**
F2	**−6.18e+11**	**−9.16e+12**
F3	**−1.14e+15**	**−8.76e+14**
F4	2.16e+02	2.16e+02
F5	**−1.30e+03**	**−1.85e+03**

superior to the LA-rBOA; because they use the selected solutions for building the Bayesian network directly; but we use them only in the learning rule of LAs to update $p(t)$; hence, for this function LA-rBOA obtains worse result. Function F5 is Schwefels problem 2.6 with global optimum on bounds. For this function, the proposed algorithm is superior to the IrBOA and the mIDEA.

Functions F5 through F25 are multi-modal functions. The results for functions F6–F25 are listed in Table 4. For each function the mean error value and the standard deviation of the obtained fitness values on the 50 independent runs are presented. Also, the statistical test between the experiments obtained with the LA-rBOA and two other algorithms for functions F6–F25 are shown in Table 5. Function F6 is shifted Rosenbrocks function, a problem between uni-modal and multi-modal function. For this function, the result for the LA-rBOA is better than the IrBOA and the mIDEA. Function F7 is shifted rotated Griewanks function without bounds, only the initialization range is given. Griewanks function is more difficult with low dimension and it is difficult to achieve the global optimum. The LA-rBOA obtains better result in comparison to the IrBOA and the mIDEA. Function F8 is shifted rotated Ackleys function with global optimum on bounds, which has a very narrow global basin and half of the dimensions of this basin are on the bounds. Hence, the search is almost like seeking a needle in a haystack. For this function, the results of the IrBOA and mIDEA are better than LA-rBOA. Functions F9 and F10 are shifted Rastrigins function and shifted rotated Rastrigins function, respectively. These two of functions have a huge number of local optima. The results demonstrate good performance of the LA-rBOA in comparison to the mIDEA and the IrBOA. Function F11 is shifted rotated Weierstrass function and Function F12 is Schwefels problem. For function F11 the mIDEA gets better

Table 4 Numerical results for the LA-rBOA, the IrBOA and the mIDEA

Function	LA-rBOA		IrBOA		mIDEA	
	Mean	Std	Mean	Std	Mean	Std
F6	**4.37e−10**	1.51e−10	6.31e−9	1.14e−9	1.75e−1	2.20e−1
F7	**5.9e−11**	1.74e−10	6.48e−9	1.46e−9	1.50e−8	1.15e−6
F8	1.98e+2	0.10e−1	**2.00e+1**	9.62e−5	2.00e+1	4.97e−4
F9	**5.17e−7**	2.56e−5	8.25e−2	1.43e+0	2.79e+1	4.56e−1
F10	**1.57e−4**	4.25e+0	4.60e−1	2.73e+0	1.17e+2	7.11e+1
F11	4.36e−3	2.10e+0	4.44e−1	7.21e+0	**3.29e−3**	3.65e+0
F12	**7.64e−13**	2.98e−10	6.8e−11	5.66e−8	4.15e+1	2.49e+1
F13	**0.04e+0**	0.34e−1	0.54e+0	0.98e−1	0.52e+1	1.32e+0
F14	**0.98e+1**	6.95e−1	1.05e+1	8.45e−1	1.38e+1	1.90e−1
F15	2.85e+2	2.34e+1	**2.44e+2**	2.12e+1	8.76e+2	1.95e+1
F16	**0.14e+1**	1.75e+1	1.35e+1	4.06e+1	7.15e+1	8.10e+1
F17	2.02e+2	0.99e+1	2.92e+3	1.44e+2	**1.56e+2**	1.58e+2
F18	**5.36e+1**	2.01e−2	7.23e+1	2.22e−2	8.30e+2	1.61e+0
F19	8.35e+2	1.32e−1	5.35e+3	2.12e+0	**8.31e+2**	1.47e+0
F20	**5.12e+0**	0.65e−1	7.32e+0	2.98e−1	8.31e+2	1.32e+0
F21	2.50e+2	0.52e−14	**1.40e+2**	1.11e−10	1.59e+2	5.44e−11
F22	**7.45e+0**	1.20e+1	3.32e+1	1.57e+1	1.56e+3	4.83e+2
F23	**1.93e+1**	1.04e−6	2.44e+1	5.42e−5	8.66e+2	8.07e−1
F24	**1.03e+1**	1.12e+2	1.53e+1	1.42e+2	9.12e+2	3.89e−1
F25	**1.52e+1**	9.21e−2	2.43e+1	3.26e−1	2.13e+2	5.50e−1

solution in compare two other algorithms but for function F12 the LA-rBOA outperforms the IrBOA and the mIDEA. As mentioned before, our test suite contains uni-modal and multi-modal functions; so, according to the obtained results it is reasonable to say that the LA-rBOA presents evidences to perform better than the IrBOA and the mIDEA.

5.4 Comparison of the LA-rBOA with other population based algorithms

In this section, we compare the performance of the LA-rBOA with some recently population based optimization algorithms: Bayesian network and Gaussian mixture model (BNGMM) [13], Variable sampling ant colony optimization algorithm (SamACO) [19], and self-adaptive global best harmony search (SGHS) [20]. BNGMM is an evolutionary continuous optimization algorithm based on Bayesian network and GMM for continuous optimization problem [13]. SamACO presents an extend version of ACO for solving continuous optimization problems. It focuses on the continuous variable sampling as a method to convert ACO from discrete optimization to continuous optimization [19]. Finally, in SGHS, a new improvisation scheme is proposed [20]. So, the good information captured in the current

Table 5 The results of statistical test for the LA-rBOA, the IrBOA and the mIDEA

Function	IrBOA	mIDEA
F6	**−1.11e+11**	**−9.04e+1**
F7	**−7.43e+10**	**−2.82e+5**
F8	4.45e+7	4.44e+7
F9	**−1.01e+0**	**−3.35e+3**
F10	**−4.51e−1**	**−5.77e−1**
F11	**−1.95e−1**	1.51e−3
F12	**−5.25e+5**	**−1.67e+0**
F13	**−1.16e+3**	**−7.40e+1**
F14	**−1.46e+1**	**−1.93e+2**
F15	1.03e+0	**−1.59e+1**
F16	**−1.55e−1**	**−2.55e−1**
F17	**−3.26e+0**	4.59e−2
F18	**−5.21e+5**	**−7.49e+3**
F19	**−2.50e+4**	4.59e+1
F20	**−5.91e+2**	**−1.18e+4**
F21	2.23e+23	7.69e+23
F22	**−1.65e+0**	**−1.66e−1**
F23	**−4.34e+0**	**−3.25e+4**
F24	**−3.82e−3**	**−1.80e+0**
F25	**−1.98e+3**	**−1.59e+4**

global best solution can be used to create new harmonies [20]. Table 6 shows the results of the LA-rBOA and the BNGMM for four chosen functions with $d = 4$. The results of the BNGMM are borrowed from [13]. From the results it can be observed that the LA-rBOA outperforms the BNGMM in all of the used functions.

Table 7 shows the obtained results for functions F6–F10 with $d = 30$ and in term of the mean error value and the standard deviation. The results of the SamACO of Table 7 are borrowed from [19]. The results show that the LA-rBOA outperforms the SamACO on the functions F6, F7 and F10.

For SGHS, the results of the mean error value and the standard deviation for five chosen function with $d = 30$ are presented in Table 8. It should be mentioned that the results of the SGHS are borrowed from [20]. Based on these results, the LA-rBOA can obtain better results in most chosen functions because of using variable structures during the run of the algorithm.

5.5 The speed of convergence of LA-rBOA

The comparison of the speed of convergence for eight chosen functions between the LA-rBOA and some related algorithms is presented in Fig. 3. Each sub-figure shows the obtained fitness value from three algorithms through the optimization process. For functions F3, F5, F9, and F10, the LA-rBOA converges faster than two other algorithms. For function F4 the mIDEA has a faster convergence rate based on

Table 6 Numerical results for the LA-rBOA and the BNGMM

Function	LA-rBOA	BNGMM
F1	**3.41e−5**	1.7e−4
F2	**3.76e−6**	5.11e−5
F6	**2.83e−9**	3.43e−7
F7	**1.11e−6**	3.03e+0

Table 7 Numerical results for the LA-rBOA and the SamACO

Function	LA-rBOA		SamACO	
	Mean	Std	Mean	Std
F6	**4.37e−10**	1.51e−10	1.26e+2	2.94e+2
F7	**5.9e−11**	1.74e−10	0.02e+0	0.01e+0
F8	1.98e+2	0.10e−1	**20.0e+0**	4.3e−3
F9	5.17e−7	2.56e−5	**1.59e−14**	2.6e−14
F10	**1.57e−4**	4.25e+0	270e+0	86.9e+0

Table 8 Numerical results for the LA-rBOA and the SGHS

Function	LA-rBOA		SGHS	
	Mean	Std	Mean	Std
F1	3.70e−12	2.50e−11	**0e+0**	0e+0
F6	**4.37e−10**	1.51e−10	21.21e+2	39.66e−2
F7	**5.9e−11**	1.74e−10	13.3e+1	13.3e+0
F9	**5.17e−7**	2.56e−5	1.86e+0	0.3e+0
F10	**1.57e−4**	4.25e+0	98.7e+0	3.0e+1

Fig. 3b. Also, for function F6, IrBOA has a faster convergence rate. It can be seen from Fig. 3, the LA-rBOA has much faster convergence rate in comparison to the two other algorithms in most of the used test functions. From the results, we can conclude that in most of the test functions the LA-rBOA has faster convergence rate; but in some of them such as function F4 and F6, the LA-rBOA has slower convergence rate. The reason of the slower convergence rate is that the LA-rBOA uses the promising solutions only in the LA learning rule. So, in some of these functions such as function F4 that has noise in fitness, the used LAs converge later than the functions that do not have noise.

5.6 The computation time of LA-rBOA

The LA-rBOA is also compared based on the computation time with the IrBOA and the mIDEA. These algorithms are implemented in MATLAB R2009a in a PC, which has a single CPU of Intel(R) Core(TM)2 Duo 3.33 GHz and a 1 GB memory. Table 9 shows the computation time for three algorithms over 50 independent runs

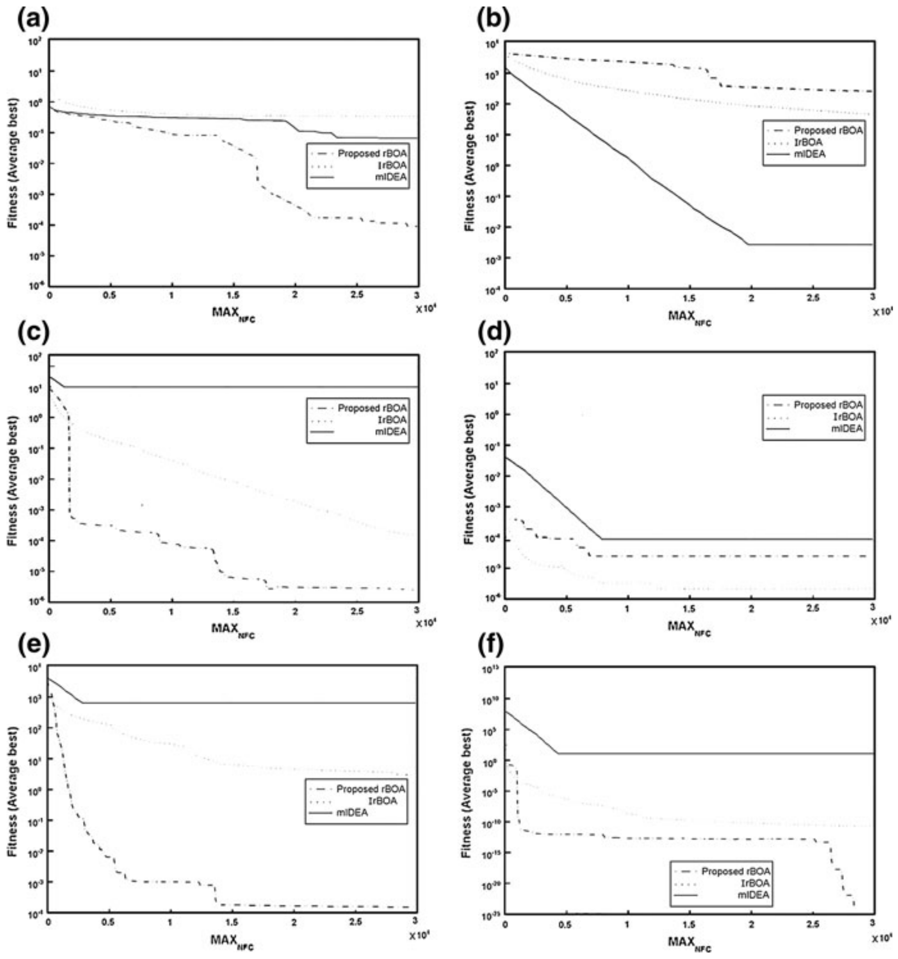

Fig. 3 The speed of convergence for the LA-rBOA, the IrBOA and the mIDEA. **a** Function *F*3. **b** Function *F*4. **c** Function *F*5. **d** Function *F*6. **e** Function *F*9. **f** Function *F*10

Table 9 Computation time (s) comparison

	F7	F8	F9	F10	F11	F12	F13
LA-rBOA	**23.40**	**21.14**	**25.52**	**88.41**	**68.30**	**78.12**	**60.19**
IrBOA	60.54	55.62	71.33	99.32	84.58	80.32	98.54
mIDEA	51.39	48.54	67.43	90.43	83.45	82.45	83.43

for seven chosen functions (F7 through F13) from the CEC2005 test suite. It can be seen that the LA-rBOA requires a smaller computation time among two other algorithms. Also, in order to show the relationship between dimensionality and scalability of the LA-rBOA, the computation time of three algorithms for functions

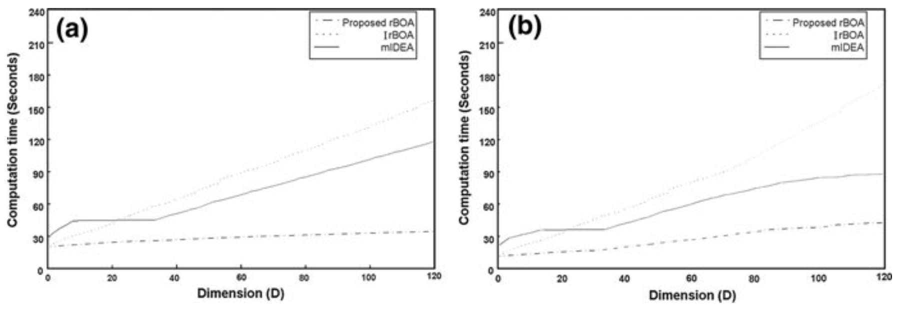

Fig. 4 The comparison of computation time with respect to different dimensionality. **a** Function *F*7. **b** Function *F*8

F7 and F8 are compared in Fig. 4. The dimension of each function is changed from 1 through 120. Based on these figures, it can be seen that the LA-rBOA has very smaller increasing rate than two other algorithms. In Fig. 4a, the LA-rBOA saves about 73 % of the computation time for function F7 in all of dimensions in comparison to the IrBOA and about 59.99 % in comparison to the mIDEA. In Fig. 4b, it can be seen that the LA-rBOA saves more computation time with increasing the dimensionality. For example in dimension 50, it saves 69.3 % of computation time in comparison to the IrBOA and 47.6 % in comparison to the mIDEA; but in dimension 120, it saves 83 % in computation time in comparison to the IrBOA and 62.5 % in comparison to the mIDEA. This may be due to the learning capability of learning automata, it also shows that the network evolves through time and a small modification of the previously learned networks can be used for the future generations.

5.7 Comparison of the LA-rBOA on the CEC2011 test functions

In this section, we compare the performance of the proposed algorithm with the performance some of the best algorithms submitted on IEEE CEC2011 competition on the CEC2011 test functions. For a fair comparison of the LA-rBOA, we evaluate the LA-rBOA with $\mathrm{Max}_{\mathrm{NFC}} = 150,000$ to the best algorithms submitted in IEEE CEC2011 on the CEC2011 test functions. Table 10 presents the comparison of the LA-rBOA with the results of the best six algorithms on the 50 independent runs. These results are borrowed from the Ref. [44]. The comparison is done based on the fitness value obtained from the LA-rBOA and the other optimization algorithms. The results show that the LA-rBOA outperforms on T01, T03, T04, T06, T07, T08, T10, T11.3, T11.7, T12.

This table shows that no algorithm outperforms other ones in all functions, but the proposed algorithm has a better result for ten functions out of 22 mentioned functions. Hence, it can be concluded that the proposed algorithm has a better results with respect to other algorithms given in this table.

Table 10 Comparison between the LA-rBOA and the best algorithms submitted on IEEE CEC2011 competition on the CEC2011 test functions

	GA-MPC	SAMODE	ENSML_DE	EA-DE-MA	Adap.DE171	ED-DE	LA-rBOA
T01	0e+0	0e+0	0e+0	1.17e−11	0e+0	0e+0	0e+0
T02	−2.84e+1	−2.84e+1	−2.14e+1	−2.84e+1	−2.84e+1	−2.84e+1	−2.20e+1
T03	1.15e−5	1.15e−5	1.15e−5	1.15e−5	1.15e−5	1.15e−5	1.15e−5
T04	13.77e+0	13.77e+0	13.80e+0	13.84e+0	14.32e+0	13.77e+0	13.77e+0
T05	−3.68e+1	−3.68e+1	−3.20e+1	−3.69e+1	−3.68e+1	−3.68e+1	−3.25e+1
T06	−29.20e+0	−29.20e+0	−19.90e+0	−29.20e+0	−29.20e+0	−29.20e+0	−29.20e+0
T07	5.00e−1	5.00e−1	12.8e−1	5.00e−1	5.00e−1	5.19e−1	5.00e−1
T08	2.20e+2	2.20e+2	2.20e+2	2.20e+2	2.20e+2	2.20e+2	2.20e+2
T09	4.67e+2	9.44e+2	7.85e+2	8.07e+0	7.58e+0	3.34e+5	5.82e+2
T10	−21.84e+0	−21.82e+0	−21.80e+0	−21.79e+0	−21.80e+0	−21.83e+0	−21.84e+0
T11.1	5.09e+4	5.13e+4	5.11e+4	18.26e+4	5.01e+4	5.13e+4	5.11e+4
T11.2	1.07e+6	1.07e+6	1.06e+6	1.97e+7	10.78e+5	1.07e+6	10.78e+5
T11.3	15.44e+3	15.44e+3	15.40e+3	15.44e+3	15.44e+3	15.44e+3	15.44e+3
T11.4	1.81e+4	1.83e+4	1.81e+4	1.88e+4	1.82e+4	1.82e+4	1.84e+4
T11.5	3.27e+4	3.28e+4	3.26e+4	3.27e+4	32.74e+4	3.28e+4	3.28e+4
T11.6	1.29e+5	1.30e+5	1.28e+5	3.34e+5	1.24e+5	1.30e+5	1.30e+5
T11.7	1.92e+6	1.92e+6	1.91e+6	1.97e+6	1.89e+6	1.90e+6	1.89e+6
T11.8	9.50e+5	9.43e+5	9.39e+5	1.22e+6	9.25e+5	9.39e+5	9.39e+5
T11.9	9.72e+5	1.01e+6	9.40e+5	1.44e+6	9.28e+5	9.93e+5	9.33e+5
T11.10	9.47e+5	9.48e+5	9.39e+5	1.22e+6	9.27e+5	9.39e+5	9.37e+5
T12	7.10e+0	6.94e+0	1.66e+1	7.97e+0	1.23e+1	9.97e+0	6.94e+0
T13	8.40e+0	8.61e+0	8.78e+0	8.63e+0	8.62e+0	1.40e+1	8.42e+0

6 Conclusion

This paper presents a new real-coded Bayesian optimization algorithm which uses a team of learning automata to construct the Bayesian network. For each possible edge in the Bayesian network, there is one learning automaton and each learning automaton tries to learn the existence or not existence of the corresponding edge. In each generation, one Bayesian network is constructed based on the action of learning automata. The next generation is sampled from the built Bayesian network and each learning automaton updates its action probability based on the quality of the built Bayesian network. The experimental results reported here show that the proposed algorithm is superior to other related algorithms based on the quality and performance measures. The proposed algorithm has two main features: (1) it builds better networks. This may be due to the learning capability of learning automata and it also shows that the network evolves through time and a small modification of the previously learned networks can be used for future generations. (2) Learning automata have a negligible computational cost for construction of the network, but it improves the quality of the network. In future works, we are trying the use learning automata as an optimization tool. We are trying to design an algorithm based learning automata for optimization and study to its behavior analytically.

Acknowledgments The authors would like to thank reviewers for their time, comments and constructive criticism, which improved the paper.

References

1. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, Reading, 1989)
2. D.E. Goldberg, K. Sastry, *Genetic Algorithms: The Design of Innovation*, 2nd edn. (Springer, Berlin, 2002)
3. J.A. Lozano, P. Larranaga, I. Inza, E. Bengoetxea (eds.), *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms* (Springer, Berlin, 2006)
4. M. Pelikan, K. Sastry, E. Cantu-Paz (eds.), *Scalable Optimization Via Probabilistic Modeling: From Algorithms to Applications* (Springer, Berlin, 2006)
5. M. Pelikan, *Bayesian Optimization Algorithm: From Single Level to Hierarchy*, Ph.D. Dissertation (University of Illinois at Urbana-Champaign, Urbana, 2002)
6. G. Harik, *Linkage Learning in Via Probabilistic Modeling in the ECGA* (University of Illinois at Urbana-Champaign, Urbana, 1999)
7. H. Muhlenbein, T. Mahnig, FDA—a scalable evolutionary algorithm for the optimization of additively decomposed functions. Evol. Comput. **7**(4), 353–376 (1999)
8. M. Pelikan, *BOA: The Bayesian Optimization Algorithm* (Springer, Berlin, 2005)
9. P. Larranaga, R. Etxeberria, J.A. Lozano, J.M. Pena, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*. Optimization in continuous domains by learning and simulation of Gaussian networks (2000), pp. 201–204
10. J. Ocenasek, J. Schwarz, in *Proceedings of the 2nd Euro-International Symposium on Computational Intelligence*. Estimation of distribution algorithm for mixed continuous discrete optimization problems (2002), pp. 227–232
11. P.A.N. Bosman, *Design and Application of Iterated Density-Estimation Evolutionary Algorithms*, Ph.D. Dissertation (Utrecht University, Utrecht, 2003)
12. C.W. Ahn, R.S. Ramakrishna, D.E. Goldberg, *Real-Coded Bayesian Optimization Algorithm: Bringing the Strength of BOA into the Continuous World,* Lecture Notes in Computer Science, vol. 3102 (Springer, Berlin, 2004), pp. 840–851

13. X. Wei, In *Proceedings of the IEEE 10th International Conference of Signal Processing (ICSP).* Evolutionary continuous optimization by Bayesian networks and Gaussian mixture model (2010), pp. 1437–1440
14. B. Moradabadi, H. Beigy, C.W. Ahn, In *Proceedings of IEEE Congress on Evolutionary Algorithm.* An improved real-coded Bayesian optimization algorithm (2011)
15. B. Moradabadi, H. Beigy, C.W. Ahn, An improved real-coded Bayesian optimization algorithm for continuous global optimization. Int. J. Innov. Comput. Inf. Control **9**(6), 2505–2519 (2013)
16. M.A.L. Thathachar, P.S. Sastry, Varieties of learning automata: an overview. IEEE Trans. Syst. Man Cybern. B Cybern. **32**, 711–722 (2002)
17. D.M. Chickering, D. Geiger, D. Heckerman, *Learning Bayesian Network is NP-hard*, Technical Report MSR-TR-94-17 (1994)
18. D. Heckerman, D. Geiger, D.M. Chickering, *A Tutorial on Learning with Bayesian Networks, Innovations in Bayesian Networks, Chapter 3* (Springer, Berlin, 2008), pp. 33–82
19. X.M. Hu, J. Zhang, H. Chung, Y. Li, O. Liu, Sam-ACO: variable sampling ant colony optimization algorithm. IEEE Trans. Syst. Man Cybern. B Cybern. **40**, 1555–1566 (2010)
20. Q. Pan, P.N. Suganthan, M.F. Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous. Appl. Math. Comput. **216**, 830–848 (2010)
21. M. Gallagher, I. Wood, J. Keith, In *Proceedings of the IEEE Congress on Evolutionary Computation*. Bayesian inference in estimation of distribution algorithms (2007), pp. 127–133
22. M. Li, D.E. Goldberg, K. Sastry, T.L. Yu, Real-coded ECGA for solving decomposable real-valued optimization problems. Link. Evol. Comput. **157**, 61–86 (2008)
23. C.W. Ahn, R.S. Ramakrishna, On the scalability of real-coded Bayesian optimization algorithm. IEEE Trans. Evol. Comput. **12**(3), 307–322 (2008)
24. D. Heckerman, D. Geiger, D.M. Chickering, *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data*, Technical Report MSR-TR-94-09 (Microsoft Research, Redmond, 1995)
25. C.W. Ahn, *Advances in Evolutionary Algorithms: Theory, Design and Practice, Studies in Computational Intelligence* (Springer, Berlin, 2006)
26. H. Beigy, M.R. Meybodi, A new continuous action-set learning automata for function optimization. J. Franklin Inst. **343**, 27 (2006)
27. G. Santharam, P.S. Sastry, M.A.L. Thathachar, Continuous action set learning automata for stochastic optimization. J. Franklin Inst. **331B**(5), 607–628 (1994)
28. B.J. Oommen, T.D. Roberts, Continuous learning automata solutions to the capacity assignment problem. IEEE Trans. Comput. **49**, 608–620 (2000)
29. 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. B Cybern. **32**, 815–820 (2002)
30. G.I. Papadimitriou, M.S. Obaidat, A.S. Pomportsis, On the use of learning automata in the control of broad-cast networks: a methodology. IEEE Trans. Syst. Man Cybern. B Cybern. **32**, 781–790 (2002)
31. H. Beigy, M.R. Meybodi, Cellular learning automata based dynamic channel assignment algorithms. Int. J. Comput. Intell. Appl. **8**(3), 287–314 (2009)
32. H. Beigy, M.R. Meybodi, Utilizing distributed learning automata to solve stochastic shortest path problems. Int. J. Uncertain Fuzz Knowl. Based Syst. **14**, 591–615 (2006)
33. O.C. Granmo, B.J. Oommen, S.A. Myrer, M.G. Olsen, Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. IEEE Trans. Syst. Man Cybern. B Cybern. **37**, 166–175 (2007)
34. H. Beigy, M.R. Meybodi, Adaptive limited fractional guard channel algorithms a learning automata approach. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **17**(6), 881–913 (2009)
35. H. Beigy, M.R. Meybodi, A learning automata-based algorithm for determination of the number of hidden units for three layer neural networks. Int. J. Syst. Sci. **40**, 101–118 (2009)
36. P.S. Sastry, G.D. Nagendra, N. Manwani, A team of continuous-action learning automata for noise-tolerant learning of half-spaces. IEEE Trans. Syst. Man Cybern. B Cybern. **40**, 19–28 (2010)
37. H. Beigy, M.R. Meybodi, Learning automata based dynamic guard channel algorithms. J. Comput. Electr. Eng. **37**(4), 601–613 (2011)
38. B.J. Oommen, M.K. Hashem, Modeling a student classroom interaction in a tutorial-like system using learning automata. IEEE Trans. Syst. Man Cybern. B Cybern. **40**, 29–42 (2010)
39. S. Narendra, K.S. Thathachar, *Learning Automata: An Introduction* (Prentice-Hall, New York, 1989)
40. K.R. Zãlik, An efficient k'-means clustering algorithm. Pattern Recogn. Lett. **29**, 1385–1391 (2008)

41. Function definitions and performance criteria for the special session on real-parameter optimization at CEC2005 (2005). http://www.ntu.edu.sg/home/EPNSugan. Accessed Apr 2013
42. Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems. http://www.ntu.edu.sg/home/EPNSugan. Accessed Apr 2013
43. D.H. Wolpert, W.G. Macready, No free lunch theorems for search. IEEE Trans. Evol. Comput. **5**(3), 295–296 (1997)
44. Competition on Testing Evolutionary Algorithms on Real-world Numerical Optimization Problems. http://www.ntu.edu.sg/home/EPNSugan. Accessed Apr 2013