Information Sciences 296 (2015) 128–146

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Hybrid learning clonal selection algorithm

Yong Peng [a], Bao-Liang Lu [a,b,*]

[a] Center for Brain-like Computing and Machine Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[b] Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

A R T I C L E   I N F O

Article history:
Received 1 February 2013
Received in revised form 22 September 2014
Accepted 26 October 2014
Available online 3 November 2014

Keywords:
Artificial immune system
Clonal selection algorithm
Baldwin effect
Baldwinian learning
Orthogonal learning
Optimization

A B S T R A C T

Artificial immune system is a class of computational intelligence methods drawing inspiration from human immune system. As one type of popular artificial immune computing model, clonal selection algorithm (CSA) has been widely used for many optimization problems. CSA mainly generates new schemes by hyper-mutation operators which simulate the immune response process. However, these hyper-mutation operators, which usually perturb the antibodies in population, are semi-blind and not effective enough for complex optimization problems. In this paper, we propose a hybrid learning clonal selection algorithm (HLCSA) by incorporating two learning mechanisms, Baldwinian learning and orthogonal learning, into CSA to guide the immune response process. Specifically, (1) Baldwinian learning is used to direct the genotypic changes based on the Baldwin effect, and this operator can enhance the antibody information by employing other antibodies' information to alter the search space; (2) Orthogonal learning operator is used to search the space defined by one antibody and its best Baldwinian learning vector. In HLCSA, the Baldwinian learning works for exploration (global search) while the orthogonal learning for exploitation (local refinement). Therefore, orthogonal learning can be viewed as the compensation for the search ability of Baldwinian learning. In order to validate the effectiveness of the proposed algorithm, a suite of sixteen benchmark test problems are fed into HLCSA. Experimental results show that HLCSA performs very well in solving most of the optimization problems. Therefore, HLCSA is an effective and robust algorithm for optimization.

1. Introduction

Artificial immune system has received increasing attention from both academic and industrial communities recently. Similar to evolutionary algorithms, artificial immune system makes use of the mechanism of vertebrate immune system to construct new intelligent algorithms, which provides some novel channels to solve optimization problems. Putting the artificial immune system into optimization becomes popular since the advance of CSA [1], which is based on the clonal selection theory and can perform multi-modal optimization while delivering good approximations for a global optimum. The main idea of clonal selection theory lies in the phenomenon that antibody can selectively react to the antigen. When

* Corresponding author at: Center for Brain-like Computing and Machine Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.
E-mail addresses: stany.peng@gmail.com (Y. Peng), bllu@sjtu.edu.cn (B.-L. Lu).

http://dx.doi.org/10.1016/j.ins.2014.10.056
0020-0255/© 2014 Elsevier Inc. All rights reserved.

an antigen is detected, the antibodies which can best recognize such antigen will proliferate by cloning. The newly cloned antibodies undergo hyper-mutations in order to increase their receptor population.

In the last several years, much effort has been made to improve the performance of CSA and consequently a large number of CSA variants were proposed. A detailed review of CSAs and their applications can be found in [2,3]. Here we will give a brief literature review of recent studies on CSA from the following three aspects: the research on theoretical analysis, computing models and applications respectively.

- Theoretical perspective. In spite of the successful applications of artificial immune system in diverse fields, there is little knowledge and hardly any theoretical investigation about how and why they perform well. In terms of convergence proofs, Villalobos-Arias et al. [4] presented a complete proof for a specific multi-objective CSA using Markov chains. Hone and Kelsey [5] pointed that a useful and valued avenue to explore into the dynamics of immune algorithms would be based on the nonlinear dynamical systems and stochastic differential equations. Timmis et al. [6] presented the detailed theoretical analysis for the three main types of artificial immune algorithms: clonal selection, immune network and negative selection algorithm. Jansen et al. analyzed different variants of immune inspired somatic contiguous hyper-mutations in [7].
- Model perspective. Many improved CSAs were proposed by modifying the operators and introducing new operators within the CSA framework. Khilwani et al. [8] proposed a fast clonal algorithm (FCA) which designs a parallel mutation operator comprising of Gaussian and Cauchy mutation strategies. In addition, a new concept was proposed for initialization, selection and clonal expansion process. Jiao et al. [9] presented a quantum-inspired immune clonal algorithm (QICA) in which the antibodies are proliferated and divided into a set of sub-populations. In QICA, the antibodies are represented by multi-state gene quantum bits. The general quantum rotation gate strategy and dynamic adjusting angle mechanism are applied to update antibody information. Gong et al. [10] proposed an orthogonal immune algorithm (OIA) based on the orthogonal initialization and a novel neighborhood orthogonal cloning operator. CSA with non-dominated neighborhood selection strategy (NNIA) [11] was proposed by Gong et al. for multi-objective optimization. Shang et al. [12] proposed an immune clonal algorithm (NICA) for multi-objective optimization problems, which makes improvements on four aspects in comparison with the traditional clonal selection computing model.

  Moreover, the search ability of CSA is enhanced by combining with other evolutionary algorithms. Gong et al. [13] put forward the differential immune clonal selection algorithm (DICSA), which uses the differential mutation and differential crossover operators. Fu et al. [14] proposed an immune algorithm-based particle swarm optimization (IA-PSO) for short-term hydropower scheduling of reservoirs, which is formulated by coupling the immune information processing mechanism with the particle swarm optimization algorithm in order to achieve a better global solution with less computational effort. Afaneh et al. [15] presented the virus detection clonal algorithm (VDC), which uses genetic algorithm to search the optimal parameters for clonal selection algorithm.
- Application perspective. Artificial immune system has been applied in diverse fields such as machine learning, pattern recognition, anomaly detection, optimization, and robotics. Liu et al. [16] incorporated the gene transposon theory into clonal selection algorithm and formulated the gene transposon CSA (GTCSA), which is used to automatically determine the number of clusters. Ma et al. [17] proposed the immune memetic clustering algorithm (IMCA), which combines the immune clonal selection and memetic algorithm simultaneously. This proposed framework is used to do image segmentation. Batista et al. [18] presented an real-coded distributed clonal selection algorithm (DCSA) for electromagnetic design optimization. Xi et al. [19] proposed an immune statistical model, which merges the statistical model and the immune algorithm together, to deal with the data analysis problems of dam displacement. Zhong et al. put artificial immune system-based computational models to applications in hyper-spectral remote sensing imagery [20–22]; specifically, an unsupervised artificial immune classifier (UAIC) was proposed to perform remote sensing image classification [20], Clonal Selection Feature Selection (CSFS) algorithm and Clonal Selection feature weighting (CSFW) algorithm were proposed to do dimensionality reduction which is formulated as an optimization problem [21] and an artificial antibody network (ABNet) based on immune network theory was designed for multi-/hyper-spectral image classification [22]. Also, CSA was used for epileptic EEG signal feature selection [23]. A detailed survey of the applications of artificial immune system can be found in [24].

Generally, the traditional clonal selection algorithm generates new schemes by various forms of hyper-mutation. These hyper-mutation operators randomly perturb a current solution, rendering the CSA a type of parallel hill climbing algorithm. Therefore, the search ability of CSA is limited when dealing with complex optimization problems. Existing studies have shown that learning mechanism can guide the evolutionary process of CSA [25]. However, only one learning rule was designed in the proposed Baldwinian learning CSA (BCSA) [25], which is less effective for complex optimization problems with different characteristics.

In this paper, we propose an enhanced CSA, hybrid learning CSA, by introducing two learning mechanisms: Baldwinian learning and orthogonal learning. In HLCSA, new schemes are generated by the way of learning from others instead of the parallel trial and error process in CSA. Different from BCSA, HLCSA maintains a strategy pool with four candidate learning rules, which assigns HLCSA the ability to adapt to different types of optimization problems. Furthermore, as described in memetic computing [26,27], a healthy balance between global search and local search can enhance the search ability of evolutionary algorithms. To improve the global search performed by Baldwinian learning, the orthogonal learning is introduced to do local search. These two learning mechanisms work together to guide the evolutionary process of CSA towards the global optimum. Obviously, HLCSA improves the traditional CSA from the model perspective.

The rest of this paper is organized as follows. Section 2 reviews the traditional CSA, the foundation of Baldwin effect and orthogonal experimental design. In Section 3, we reveal the inefficacy of traditional CSA, present the two learning mechanisms (Baldwinian learning and orthogonal learning) to enhance the performance of CSA, and formulate the whole framework of our proposed HLCSA. Section 4 presents the benchmark test functions, the experimental settings for each algorithm, the experimental results as well as several discussions. Conclusions and future work are given in Section 5.

2. Preliminaries

In this section, we review the traditional CSA, the foundation of Baldwin effect and orthogonal experimental design (OED) theory, which will form the basis for introducing our proposed HLCSA framework.

2.1. Clonal selection algorithm

Before introducing the basic flow of CSA, we present some immunological terms first for helping to understand this algorithm.

- **Antigen**. In immunology, any substance can be called antigen if it can cause the immune systems to produce antibodies against its invasion. In evolutionary computation, antigens refer to the pending problems. Taking the following optimization problem for example,

$$\min\ f(\vec{x}), \qquad \vec{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{S} = \prod_{i=1}^{D}[a_i, b_i], \tag{1}$$

  where $f(\vec{x})$ is the objective function which is also called antigen, $\vec{x} \in \mathbb{S}$ is a variable vector, $\mathbb{S} \in \mathbb{R}^D$ is the feasible space and $\forall i \in \{1, 2, \ldots, D\}, -\infty < a_i < b_i < +\infty$.
- **Antibody**. An antibody (Ab), also known as an immunoglobulin (Ig), is a large Y-shaped protein produced by B-cells that is used by the immune system to identify and neutralize foreign objects such as bacteria and viruses. In this paper, an antibody is the representation of a candidate solution of an antigen. The antibody $\vec{a} = (a_1, a_2, \ldots, a_D)$ is the coding of variable $\vec{x}$, denoted by $\vec{a} = e(\vec{x})$ and $\vec{x}$ is called the decoding of antibody $\vec{a}$, expressed as $\vec{x} = e^{-1}(\vec{a})$. The representation of antibody $\vec{a}$, which may be different from that of antigen, can be binary string, real number sequence, symbolic sequence and character. In this paper, we adopt real-valued representation, which is simply $\vec{a} = e(\vec{x}) = \vec{x}$.
  Let $\mathbf{I}$ be the antibody space, where $\vec{a} \in \mathbf{I}$. The antibody population $\mathbf{A} = \{\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n\}$, where $\vec{a}_i \in \mathbf{I}, 1 \leqslant i \leqslant n$, is an n-dimensional group of antibody $\vec{a}$, where the positive integer n is the antibody population size.
- **Affinity**. In immunology, affinity is the fitness measurement of an antibody. For the optimization problem defined in (1), the affinity $F(\cdot)$ is a certain type of function of the value of objective function $f(\cdot)$. In this paper, we use the identity function for simplicity, which is $F(\cdot) = f(\cdot)$.
  Based on the terminologies above, we introduce the basic flow of CSA below, whose framework is described in Algorithm 1.

Algorithm 1. Clonal Selection Algorithm

```
Step 1. Initialization: Determine the parameters and termination criterion; Randomly generate the initial antibody
    population $\mathbf{A}(0)$; Set $t = 0$;
Step 2. Evaluation: Calculate the affinity of each antibody in $\mathbf{A}(t)$;
Step 3. Clonal Proliferation: Generate $\mathbf{X}(t)$ by applying clonal proliferation operator to $\mathbf{A}(t)$;
Step 4. Hyper-mutation: Generate $\mathbf{Y}(t)$ by applying hyper-mutation operator to $\mathbf{X}(t)$;
Step 5. Evaluation: Calculate the affinity of each antibody in $\mathbf{Y}(t)$;
Step 6. Clonal Selection: Generate $\mathbf{A}(t+1)$ by applying clonal selection operator to $\mathbf{Y}(t)$ and $\mathbf{A}(t)$;
Step 7. Termination Test: If termination criterion is met, stop and output the antibody with the highest affinity in
    $\mathbf{A}(t+1)$; Otherwise, $t = t + 1$, go to Step 3.
```

The three main operators, clonal proliferation, hyper-mutation and clonal selection, are explained as follows.

- **Clonal Proliferation** T^C. In immunology, clone means asexual propagation, so that a group of identical cells can be descended from a single ancestor. In the t-th generation of CSA, the resulting population can be obtained by applying the clonal proliferation operator T^C to the population $\mathbf{A}(t)=\{\vec{a}_1(t), \vec{a}_2(t), \ldots, \vec{a}_n(t)\}$:

$$\mathbf{X}(t) = T^C(\mathbf{A}(t)) = \left\{T^C(\vec{a}_1(t)), T^C(\vec{a}_2(t)), \ldots, T^C(\vec{a}_n(t))\right\}, \tag{2}$$

where $\mathbf{X}_i(t) = T^C(\vec{a}_i(t)) = \left\{\vec{x}_{i1}(t), \vec{x}_{i2}(t), \ldots, \vec{x}_{iq_i}(t)\right\}$, and $\vec{x}_{ij}(t) = \vec{a}_i(t)$, $i = 1,2,\ldots,n, j = 1,2,\ldots,q_i$. q_i is a self-adaptive parameter or a constant, termed *clonal scale* or *clonal factor*. Essentially, clonal proliferation on antibody $\vec{a}_i(t)$ is to make q_i identical copies of $\vec{a}_i(t)$.

- **Hyper-mutation** T^M. In immunology, hyper-mutation is the main mechanism for the immune system to recognize external pattern in the form of antibody gene mutation and compilation so as to gain higher affinity [28]. Inspired by immune response process, the hyper-mutation operation T^M exploits local areas around antibodies by introducing blind perturbation and this is the only operation in traditional CSA to generate new schemes. When applying hyper-mutation operator T^M on population $\mathbf{X}(t)$, the resulting population is

$$\mathbf{Y}(t) = T^M(\mathbf{X}(t)) = \left\{T^M(\mathbf{X}_1(t)), T^M(\mathbf{X}_2(t)), \ldots, T^M(\mathbf{X}_n(t))\right\}, \tag{3}$$

where $\mathbf{Y}_i(t) = \left\{\vec{y}_{i1}(t), \vec{y}_{i2}(t), \ldots, \vec{y}_{iq_i}(t)\right\}$ and $\vec{y}_{ij}(t) = T^M(\vec{x}_{ij}(t))$, $i = 1,2,\ldots,n, j = 1,2,\ldots,q_i$. $T^M(\vec{x}_{ij}(t))$ means modifying each element of antibody $\vec{x}_{ij}(t)$ based on a general mutation operator with probability P_m, so each antibody in the population $\mathbf{X}(t)$ in each generation will undergo about $D \times P_m$ mutations, where D is the dimension of variable vector.

- **Clonal Selection** T^S. For $\forall i = 1,2,\ldots,n$, if $\vec{y}_i^*(t)$ is the antibody with the highest affinity in $\mathbf{Y}_i(t)$, then the process of applying T^S on $\mathbf{Y}_i(t)$ is

$$\vec{a}_i(t+1) = T^S(\mathbf{Y}_i(t) \cup \vec{a}_i(t)) = \begin{cases} \vec{y}_i^*(t), & \text{if } F\left(\vec{y}_i^*(t)\right) > F(\vec{a}_i(t)), \\ \vec{a}_i(t), & \text{otherwise.} \end{cases} \tag{4}$$

The population entering the next generation is

$$\begin{aligned}\mathbf{A}(t+1) = T^S(\mathbf{Y}(t) \cup \mathbf{A}(t)) &= \left\{T^S(\mathbf{Y}_1(t) \cup \vec{a}_1), T^S(\mathbf{Y}_2(t) \cup \vec{a}_2), \ldots, T^S(\mathbf{Y}_n(t) \cup \vec{a}_n)\right\} \\ &= \{\vec{a}_1(t+1), \vec{a}_2(t+1), \ldots, \vec{a}_n(t+1)\}\end{aligned} \tag{5}$$

This selection process can be seen as a sort of parallel hill climbing and each antibody is locally optimized via affinity maturation process.

2.2. Baldwin effect

Baldwin effect is a theory of a possible evolutionary process. Within this framework, selected offspring would tend to have an increased capacity for learning new skills rather than being confined to genetically coded, relatively fixed abilities. In Baldwin effect, it places emphasis on the fact that the sustained behavior of a specie or group can shape the evolution of such specie. Its underlying idea is that enhancing the evolutionary process through learning the models of Baldwin effect are diverse.

The basic theory of Baldwin effect is twofold: (1) Learning can improve the individual performance, which will be reflected by its phenotype; and (2) the ability obtained from learning will be finally integrated into the genotype and be inherited in following evolutionary process. Based on the Baldwin effect, an individual will survive longer if its learned fitness is better and consequently it will be replaced by a lower probability in the evolutionary process. If this individual can survive for a sufficient number of generations, it will be possible to evolve, by genetic operators, into the right genotype corresponding to the learned fitness [29]. Even though the characteristics to be learned in the phenotype space were not genetically specified, there is evidence [30,31] to show that the Baldwin effect is able to direct the genotypic changes.

Recently, Baldwin effect was widely incorporated into evolutionary computing models to improve their performance. Yuan et al. [32] presented a hybrid genetic algorithm (HGA) in which the local search step is based on the Baldwin effect. Zhang [33] proposed a sexual adaptive genetic algorithm (AGA). By employing the Baldwin effect, AGA guides individuals to forward or reverse learning and enables the transmission of fitness information between parents and offspring to adapt individuals' acquired fitness. Gong et al. [34,25] first introduced the Baldwin effect into CSA and formulated the Baldwinian clonal selection algorithm (BCSA), which guides the evolution of each antibody by the differential information of other antibodies in the population. Qi et al. [35] incorporated the Baldwinian learning strategy into the nondominated neighborhood immune algorithm for multi-objective optimization. The Baldwinian learning strategy extracts the evolving environment of current population by building a probability distribution model and generates a predictive improving direction by combining the environment information and evolving history of the parental individuals. Zhang et al. [36] incorporated the Larmarckian and Baldwinian learning by analyzing the search ability between exploration and exploitation in the process of migration among subpopulations as well as in the hybridization of differential evolution (DE) [37] and local search.

2.3. Orthogonal experimental design

Considering a system whose cost depends on K factors (i.e., variables) and each factor can take one of Q levels (i.e., values), we aim at finding the best combination of levels for each factor to minimize the system cost. The simplest way is to do one experiment for each combination of factor levels and then select the best one. Obviously, it is impossible to try all Q^K trials when K and Q are large. As an alternative, experimental design methods provide us an effective way to deal with this combinational optimization problem, which can sample a small number of well representative combinations for testing.

Orthogonal design is one of several widely used experimental design tools, which works on a predefined table, called *orthogonal array* (OA). An OA with K factors and Q levels per factor is denoted by $L_M(Q^K)$, where L denotes the orthogonal array and M is the number of combinations of test cases. As an example, $L_9(3^4)$ is shown as follows:

$$L_9(3^4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 3 & 3 & 2 & 1 \end{bmatrix}. \tag{6}$$

Each row in this array is a combination of levels, that is, an experiment. So there are 9 experiments in total. Specifically, the fourth row [2 1 2 3] means that factors 1, 2, 3, 4 are respectively set as levels 2, 1, 2 and 3.

The orthogonality of an OA means that: (1) Each level of the factor occurs equal times in each column and (2) each possible level combination of any two given factors occurs equal times in the array. Orthogonal experimental design is a sampling method used in the predefined space, which forms the orthogonal learning strategy.

3. Hybrid learning clonal selection algorithm

In this section, two learning mechanisms, the Baldwinian learning and orthogonal learning, will be described in detail. Based on these two important ingredients, the whole framework of HLCSA as well as several explanations will be presented.

3.1. Baldwinian learning

The Baldwinian learning draws inspiration from the Baldwin effect, which basically encourages guiding the evolutionary process through learning [38,39]. It makes use of the exploration performed by the phenotype to facilitate the evolutionary search for good genotypes. The Baldwinian learning mechanism was first introduced to improve the performance of CSA by Gong et al. [25]. The Baldwinian learning is essentially a ‘rand/1’ like operator, which is the widely used operation in differential evolution [37]. In the resulting Baldwinian learning CSA (BCSA), each antibody is enhanced by learning from the differential information of two randomly selected antibodies according to user-defined probability. However, [40,41] show that learning from exemplar, which are corresponding to ‘best/2’ and ‘best/1’ operators, may have some advantages over ‘rand/1’. Furthermore, some other learning rules may be suitable for some specific problems. Wang et al. proposed the composite differential evolution (CoDE) model [42], which also uses composite trial vector generating strategies.

Obviously, different learning strategies have different characteristics which will be helpful for different optimization problems. To this end, we maintain a strategy candidate pool which contains several effective learning rules instead of using only a single learning rule. In the present study, the newly designed Baldwinian leaning pool maintains the following four strategies:

- BL/rand/1:

$$\vec{y}_{i,G} = \vec{x}_{r1,G} + s \cdot (\vec{x}_{r2,G} - \vec{x}_{r3,G}); \tag{7}$$

- BL/rand/2:

$$\vec{y}_{i,G} = \vec{x}_{r1,G} + s \cdot (\vec{x}_{r2,G} - \vec{x}_{r3,G}) + s \cdot (\vec{x}_{r4,G} - \vec{x}_{r5,G}); \tag{8}$$

- BL/current-to-rand/1:

$$\vec{y}_{i,G} = \vec{x}_{i,G} + rand \cdot (\vec{x}_{r1,G} - \vec{x}_{i,G}) + s \cdot (\vec{x}_{r2,G} - \vec{x}_{r3,G}); \tag{9}$$

- BL/current-to-best/2:

$$\vec{y}_{i,G} = \vec{x}_{i,G} + s \cdot (\vec{x}_{best,G} - \vec{x}_{i,G}) + s \cdot (\vec{x}_{r1,G} - \vec{x}_{r2,G}) + s \cdot (\vec{x}_{r3,G} - \vec{x}_{r4,G}); \tag{10}$$

where $\overrightarrow{x}_{i,G}$ is the selected antibody to do Baldwinian learning and $\overrightarrow{y}_{i,G}$ is the resulting antibody. $r1, r2, r3, r4$ and $r5$ are different indices uniformly randomly selected from $\{1, 2, \ldots, \mathrm{NP}\} \setminus \{i\}$, s is the strength of Baldwinian learning, and *rand* is a random number in $[0, 1]$.

When performing Baldwinian learning in each generation, each learning strategy in the candidate pool is used to create a new vector. Thus, four vectors are generated for each antibody and then the best one will be selected to enter the next stage.

The reasons accounting for selecting these four learning rules to construct the strategy pool are as follows. In the 'BL/rand/1' strategy, all vectors are selected randomly from the population and have no bias to any specific direction. This strategy usually demonstrates slow convergence speed and bears stronger exploration capacity. Therefore, it is usually more suitable for solving multi-modal problems than the strategies relying on the best solution found so far. Similarly, the 'BL/rand/2' strategy, which is a two-difference-vectors-based strategy, may lead to better perturbation than the strategies with only one difference vector and thus can generate more different vectors than 'BL/rand/1' strategy. It helps to maintain the diversity of population and avoid premature, which can avoid the slow convergence problem in later evolutionary process to some extent. 'BL/current-to-rand/1' is a rotation-invariant strategy [43] and will be beneficial to the rotated multi-modal problems. Strategies such as 'best/1' and 'best/2', which rely on the best solution found so far, usually lead to fast convergence speed and perform well when solving unimodal problems and easily get stuck in local optima. Therefore we use the 'current-to-best/2' strategy instead and this two-difference-vectors-based strategy can bring much diversity to avoid premature when solving multi-modal problems.

3.2. Orthogonal learning

After performing the Baldwinian learning on each antibody, the antibody as well as its corresponding best Baldwinian learning vector are obtained. We randomly select several pairs of them to undergo orthogonal learning (OL), which means the OL operator is used to probe the hyper-rectangle defined by the antibody and its best Baldwinian learning vector. The detailed implementation of OL strategy will be described later. The OED is previously prepared to reduce the computational effort in each generation; however, within our framework, following the idea in [44], we only apply OL on one pair of these antibodies to save the computational cost and keep the implementation simple. The experimental results show that the sampling solutions produced by OL strategy can effectively probe such defined search space.

The motivations of introducing OL strategy in this study are twofold.

- The Baldwinian learning mechanism performs as exploration by using the phenotype to guide the evolutionary search for good genotypes. However, as mentioned in [26], the excellent search ability relies on a healthy balance in exploration (global search) and exploitation (local refinement) under a fixed computational budget, so we need to incorporate a local improvement mechanism to enhance the evolutionary search efficacy of CSA. Accordingly, OL strategy here can be seen as a local search strategy for compensating the global search strategy performed by Baldwinian learning.
- Recent studies have shown that incorporating orthogonal experimental design in evolutionary computing algorithms can improve their performance significantly. Zhang and his collaborators designed the orthogonal crossover (OX) based on OED [45,46]. OX operators can make a systematic and statistically sound search in a region defined by parental solutions. Similarly, OED has been widely incorporated into other evolutionary models such as CSA [10], differential evolution [47,44], particle swarm optimization [48,49] and effectively improve the performance of these models. Furthermore, orthogonal design was used to solve constrained evolutionary optimization problem [50]. In this paper, we use the OED to form an OL strategy, which searches the specified hyper-rectangle in order to construct some more promising individuals. From this perspective, OL can be viewed as a strategy to refine solutions.

Here we give the detailed description of OL strategy. If $\overrightarrow{a} = (a_1, a_2, \ldots, a_D)$ and $\overrightarrow{b} = (b_1, b_2, \ldots, b_D)$ are the two selected antibodies to undergo OL search. The search range for i-th dimension is defined as $[\min(a_i, b_i), \max(a_i, b_i)]$.

Considering the fact that the dimension of variables D may be larger than K, we use the quantization technique [46] to partition $(x_1, x_2, \ldots, x_D)$ into K subvectors, specifically

$$\begin{cases} \overrightarrow{F}_1 = (x_1, \ldots, x_{t_1}) \\ \overrightarrow{F}_2 = (x_{t_1+1}, \ldots, x_{t_2}) \\ \cdots \\ \overrightarrow{F}_K = (x_{t_{K-1}+1}, \ldots, x_D) \end{cases} \tag{11}$$

where $t_1, t_2, \ldots, t_{K-1}$ are randomly generated integers which satisfy $1 < t_1 < t_2 < \cdots < t_{K-1} < D$. OL treats each $\overrightarrow{F}_i, i = 1, 2, \ldots, K$, as a factor and accordingly there are K factors in total. Based on the definition of $L_M(Q^K)$, we should define Q levels for $\overrightarrow{F}_i$:

$$\begin{cases} \overrightarrow{L}_{i1} = (l_{t_{i-1}+1,1}, l_{t_{i-1}+2,1}, \ldots, l_{t_i+1,1}) \\ \overrightarrow{L}_{i2} = (l_{t_{i-1}+1,2}, l_{t_{i-1}+2,2}, \ldots, l_{t_i+1,2}) \\ \ldots \\ \overrightarrow{L}_{iQ} = (l_{t_{i-1}+1,Q}, l_{t_{i-1}+2,Q}, \ldots, l_{t_i+1,Q}) \end{cases} \tag{12}$$

and

$$l_{i,j} = \min(a_i, b_i) + \frac{j-1}{Q-1} \cdot [\max(a_i, b_i) - \min(a_i, b_i)], \tag{13}$$

where $j = 1, 2, \ldots, Q$. In this way, $L_M(Q^K)$ can be used on factors $\overrightarrow{F}_i, i = 1, 2, \ldots, K$, to construct M solutions, where each factor holds Q levels.

If the dimension of variables D is smaller than K, the first D columns of the orthogonal array can be used to OL directly. If $L_9(3^4)$ is used on 2-dimensional space (the first 2 columns of array (6) are used) and two parental solutions are $\overrightarrow{a}$=(-1,2) and $\overrightarrow{b}$=(0,1), the corresponding nine quantized points generated by orthogonal learning are shown in Fig. 1.

The concrete examples of implementing this orthogonal learning with quantization can be found in [46,44].

3.3. The proposed algorithm framework

This section introduces the proposed hybrid learning clonal selection algorithm (HLCSA), whose overall framework is summarized in Algorithm 2.

Algorithm 2. Hybrid Learning Clonal Selection Algorithm (HLCSA)

```
Parameters
    NP: population size;
    q: clonal scale (=4);
    s: strength of Baldwinian learning;
    $L_M(Q^K)$: orthogonal array;
    mFES: maximum number of function evaluations;
    FES: current number of function evaluations
Step 1: Initialization
    Set current generation number $t = 0$ and current number of function evaluations FES = 0;
    Randomly generate initial population in the feasible solution space:
        $\mathbf{A}(0)=\left\{\overrightarrow{a}_1(0), \overrightarrow{a}_2(0), \ldots, \overrightarrow{a}_{NP}(0)\right\}$;
    Calculate the affinity of each antibody $\overrightarrow{a}_i$ ($i = 1, 2, \ldots, NP$) in the initial population;
    Set the current number of function evaluations FES = NP;
Step 2: Randomly choose an index $k$ from $\{1, 2, \ldots, NP\}$;
Step 3: Clonal Proliferation
    Apply clonal proliferation operator $T^C$ on $\mathbf{A}(t)$ and generate the resulting population $\mathbf{X}(t)$ (each antibody in $\mathbf{A}(t)$ will
  be proliferated to four antibodies);
Step 4: Baldwinian Learning
    Apply Baldwinian learning strategy on $\mathbf{X}(t)$ and generate the resulting population $\mathbf{Y}(t)$ (each antibody in $\mathbf{X}(t)$ will
  undergo Baldwinian learning according to (7)–(10));
    Choose the best one of the four Baldwinian learning vectors originating from the same antibody and form a new
  population $\mathbf{Z}(t)$;
    Update the number of function evaluations as FES =FES + q·NP;
Step 6: Orthogonal Learning
    Apply OL strategy to search the space defined by $\overrightarrow{a}_k(t)$ and $\overrightarrow{z}_k(t)$, which will generate $M$ sampling solutions;
    Choose the best one of these $M$ solutions to substitute $\overrightarrow{z}_k(t)$;
    Update the number of function evaluations as FES =FES+$Q^2$;
Step 7: Clonal Selection
    Generate $\mathbf{A}(t+1)$ by applying clonal selection operator $T^S$ on $\mathbf{Z(t)}$ and $\mathbf{A}(t)$;
Step 8: If FES $\geqslant$ mFES, stop and output the best results; otherwise, set $t = t + 1$ and go to Step 2.
```

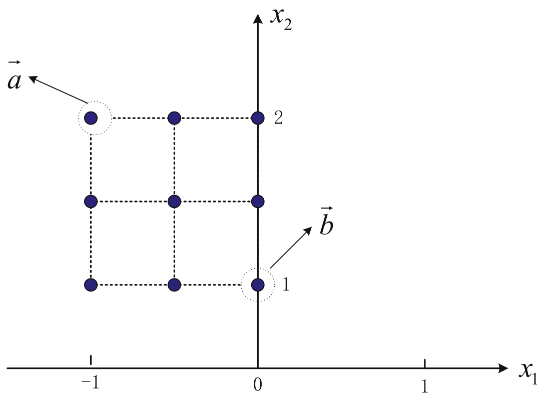

Fig. 1. Illustration of quantization in two-dimensional space.

Below are explanations of some operations defined in the HLCSA framework.

- There are four strategies in the candidate pool for Baldwinian learning and thus each antibody will be proliferated to four antibodies. Therefore the *clonal scale q* is four, which means that there will be 4∗NP antibodies after Baldwinian learning if the initial population size is NP. This process can be described as follows:

$$\mathbf{X}(t) = \left\{ \underbrace{\vec{x}_{11}(t), \ldots, \vec{x}_{1q}(t)}_{q}, \underbrace{\vec{x}_{21}(t), \ldots, \vec{x}_{2q}(t)}_{q}, \ldots, \underbrace{\vec{x}_{\mathrm{NP},1}(t), \ldots, \vec{x}_{\mathrm{NP},q}(t)}_{q} \right\}$$
$$= \left\{ \underbrace{\vec{a}_{1}(t), \ldots, \vec{a}_{1}(t)}_{4}, \underbrace{\vec{a}_{2}(t), \ldots, \vec{a}_{2}(t)}_{4}, \ldots, \underbrace{\vec{a}_{\mathrm{NP}}(t), \ldots, \vec{a}_{\mathrm{NP}}(t)}_{4} \right\},$$

$$\mathbf{Y}(t) = \left\{ \underbrace{\vec{y}_{11}(t), \ldots, \vec{y}_{1q}(t)}_{q}, \underbrace{\vec{y}_{21}(t), \ldots, \vec{y}_{2q}(t)}_{q}, \ldots, \underbrace{\vec{y}_{\mathrm{NP},1}(t), \ldots, \vec{y}_{\mathrm{NP},q}(t)}_{q} \right\},$$

where $\vec{y}_{ij}$ is obtained from $\vec{x}_i$ based on (7)–(10) for $i = 1, 2, \ldots, \mathrm{NP}$ and $j = 1, 2, \ldots, q$. Among these four antibodies, the best one is selected and consequently NP antibodies are selected to form an intermediate population $\mathbf{Z}(t)$ as follows:

$$\mathbf{Z}(t) = \left\{ \vec{y}_{1,best}(t), \vec{y}_{2,best}(t), \ldots, \vec{y}_{NP,best}(t) \right\} \triangleq \left\{ \vec{z}_{1}(t), \vec{z}_{2}(t), \ldots, \vec{z}_{NP}(t) \right\},$$

where $F(\vec{z}_i(t)) = \min\left\{ F(\vec{y}_{i1}(t)), \ldots, F(\vec{y}_{iq}(t)) \right\}$.
- Once choosing an index from [1, NP] randomly, we will obtain two parental antibodies, which are antibody $\vec{a} \in \mathbf{A}(t)$ and its best Baldwinian learning vector $\vec{z} \in \mathbf{A}(t)$. Applying OL strategy on this hyper-rectangle defined by $\vec{a}$ and $\vec{z}$, this operation will generate M antibodies if $L_M(Q^k)$ OED [45] used. Choosing the best one of these M new solutions to substitute $\vec{z}$. The following step is clonal selection between $\mathbf{A}(t)$ and $\mathbf{Z}(t)$.
- As mentioned in Section 2.1, the hyper-mutation is a semi-blind operator and in HLCSA its role has been replaced by Baldwininan learning and orthogonal learning. Therefore, there is no hyper-mutation operation in the proposed HLCSA framework.

4. Experimental studies

In this section, we conduct experiments to evaluate the performance of HLCSA by solving 16 commonly used global optimization problems. We firstly compare HLCSA with traditional CSA and then compare HLCSA with some state-of-the-art evolutionary computing models. Some discussions on the performance analysis of HLCSA are also included.

4.1. Test functions

Table 1 gives the test functions used in the experiments. They can be categorized into four types: $f_1 \sim f_2$ are unimodal functions, $f_3 \sim f_8$ are unrotated multi-modal functions, $f_9 \sim f_{14}$ are rotated multi-modal functions, and $f_{15} \sim f_{16}$ are composite functions. The detailed characteristics of these functions can be found in [51,52].

Table 1
Benchmark functions used in our experimental study.

Name	Test function	D	S	$f_{\min}$
Sphere fun	$f_1(x) = \sum_{i=1}^{D} x_i^2$	10/30	$[-100, 100]$	0
Rosenbrock's fun	$f_2(x) = \sum_{i=1}^{D-1}\left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	10/30	$[-2.048, 2.048]$	0
Ackley's fun	$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	10/30	$[-32.768, 32.768]$	0
Griewanks's fun	$f_4(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	10/30	$[-600, 600]$	0
Weierstrass fun	$f_5(x) = \sum_{i=1}^{D}\left\{\sum_{k=1}^{kmax}[a^k\cos(2\pi b^k(x_i + 0.5))]\right\} - D\sum_{k=1}^{kmax}\left\{a^k\cos(2\pi b^k \cdot 0.5)\right\}$ $a = 0.5, b = 3, kmax = 20$	10/30	$[-0.5, 0.5]$	0
Rastrigin's fun	$f_6(x) = \sum_{i=1}^{D}\{x_i^2 - 10\cos(2\pi x_i) + 10\}$	10/30	$[-5.12, 5.12]$	0
Noncont. Ras	$f_7(x) = \sum_{i=1}^{D}\{y_i^2 - 10\cos(2\pi y_i) + 10\}$, $y_i = \begin{cases} x_i & \lvert x_i\rvert < 0.5 \\ \frac{round(2x_i)}{2} & \lvert x_i\rvert \geqslant 0.5 \end{cases}, i = 1, 2, \ldots, D$	10/30	$[-5.12, 5.12]$	0
Schwefel's fun	$f_8(x) = 418.9829D - \sum_{i=1}^{D}\{x_i\sin(\lvert x_i\rvert^{0.5})\}$	10/30	$[-500, 500]$	0
Rot. Ackley's	$f_9(x) = f_3(y), y = M * x$	10/30	$[-32.768, 32.768]$	0
Rot. Griewanks's	$f_{10}(x) = f_4(y), y = M * x$	10/30	$[-600, 600]$	0
Rot. Weierstrass	$f_{11}(x) = f_5(y), y = M * x$	10/30	$[-0.5, 0.5]$	0
Rot. Rastrigin's	$f_{12}(x) = f_6(y), y = M * x$	10/30	$[-5.12, 5.12]$	0
Rot. noncon Ras	$f_{13}(x) = f_7(y), y = M * x$	10/30	$[-5.12, 5.12]$	0
Rot. Schwefel's	$f_{14}(x) = 418.9829D - \sum_{i=1}^{D} z_i$, $z_i = \begin{cases} y_i\sin(\lvert y_i\rvert^{0.5}) & \lvert y_i\rvert \leqslant 500 \\ 0.001(\lvert y_i\rvert - 500)^2 & \lvert y_i\rvert > 500 \end{cases},$ $i = 1, 2, \ldots, D; y = M * (x - 420.96) + 420.96$	10/30	$[-500, 500]$	0
Composition	$f_{15} = CF1$	10/30	$[-5, 5]$	0
Composition	$f_{16} = CF2$	10/30	$[-5, 5]$	0

4.2. Experimental settings

The experiments are divided into two parts: (1) We compare the HLCSA with the traditional clonal selection algorithm (CLONALG) [1] on the above mentioned 16 test problems with 10 dimensions and 30 dimensions, respectively and (2) experiments are conducted on the proposed HLCSA model and other seven state-of-the-art evolutionary algorithms on the same optimization problems. The seven representative evolutionary algorithms are listed as follows:

- Baldwinian clonal selection algorithm (BCSA) [25];
- Self-adaptive differential evolution (SaDE) [43];
- Orthogonal crossover based differential evolution (OXDE) [44];
- Orthogonal genetic algorithm with quantization (OGA/Q) [46];
- Comprehensive learning particle swarm optimization (CLPSO) [51];
- Evolutionary strategy with covariance matrix adaptation (CMA-ES) [53];
- Global and local real-coded genetic algorithm (GL-25) [54].

The population size is set as 30 for both 10-D and 30-D problems and the maximum number of function evaluations (mFES) are set as $10,000 \times D$, respectively. All experiments were run 30 independent times. Any specific algorithm-related parameters are set exactly the same as those in the original work. The results including the mean values and standard deviations are reported. In HLCSA, the parameter s, which is the strength of Baldwinian learning, is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by $N(0.5, 0.3)$. Different values of s are randomly sampled from this normal distribution and applied to each Baldwinian learning rule. In OGA/Q [46], the initial population was generated with orthogonal design. In this paper, the global optimal solutions of some test instances are located at the center of the feasible solution space. If the orthogonal design is used to generate the initial population, it can directly find the global optimum after initialization without any evolutionary process. Therefore, we report the experimental results as well as the convergence of OGA/Q based on random initialization for fair comparison.

4.3. Experimental results and discussions

4.3.1. Results of HLCSA and CLONALG

Table 2 shows the statistical results of CLONALG and HLCSA in optimizing the 16 test problems with $D = 10$ based on 30 independent runs including the maximum, minimum, mean and standard deviation. The best mean values are shown in bold face. From the results, we can observe that, for all these test instances, HLCSA performs much better than CLONALG. HLCSA can find the exact global optima of functions $f_4, f_5, f_6, f_7, f_8, f_{11}, f_{14}$ and f_{15} with probability 1 and the approximate global optima of functions f_2, f_3 and f_9. In fact, from the statistical results we know HLCSA can find the global optimum in most of cases for function 16 only with small probability being trapped in local optima. This shows that the learning mechanism

Table 2
Results of traditional clonal selection algorithm and HLCSA when $D = 10$.

ALGs	Clonal selection algorithm				Hybrid learning clonal selection algorithm			
	Max	Min	Mean	Std	Max	Min	Mean	Std
f_1	1.6739e−007	2.2182e−009	5.0414e−008	4.4098e−008	5.4303e−052	2.8538e−055	**4.2228e−053**	9.9892e−053
f_2	8.1888e+000	3.1046e−001	5.5057e+000	2.7914e+000	2.9934e−027	0	**3.9087e−028**	6.6496e−028
f_3	4.7052e−002	5.0332e−005	2.8532e−003	8.5712e−003	2.6645e−015	0	**2.5757e−015**	4.8648e−016
f_4	6.6558e−002	1.2522e−007	2.6092e−002	1.9092e−002	0	0	**0**	0
f_5	2.2506e−002	6.2798e−003	1.2569e−002	4.4216e−003	0	0	**0**	0
f_6	1.3129e+001	3.2257e+000	8.4141e+000	2.4848e+000	0	0	**0**	0
f_7	8.0000e+000	3.0000e+000	5.7681e+000	1.3571e+000	0	0	**0**	0
f_8	5.9346e+002	1.9237e+002	3.6436e+002	9.7648e+001	0	0	**0**	0
f_9	2.5278e+000	9.1471e−003	1.0619e+000	7.9728e−001	3.5527e−015	3.5527e−015	**3.5527e−015**	0.0000e−000
f_{10}	6.3623e−001	1.6119e−001	4.2974e−001	1.0264e−001	6.8888e−002	0	**2.8802e−002**	1.9129e−002
f_{11}	8.2950e+000	2.1579e+000	6.0837e+000	1.5496e+000	0	0	**0**	0
f_{12}	5.0439e+001	3.0548e+001	4.2193e+001	5.4173e+000	9.9596e+000	0	**4.2783e+000**	2.0095e+000
f_{13}	5.2738e+001	3.1727e+001	4.3672e+001	5.5570e+000	1.1219e+001	5.4104e−011	**4.2442e+000**	2.6469e+000
f_{14}	2.2228e+003	9.4657e+002	1.9305e+003	2.3358e+002	0	0	**0**	0
f_{15}	1.0706e+002	1.4519e+000	4.5354e+001	3.2450e+001	0	0	**0**	0
f_{16}	8.4258e+001	1.7281e+001	5.1701e+001	1.7347e+001	2.4468e+000	0	**4.6177e−001**	8.1247e−001

is appealing in searching new schemes within CSA than the widely used hyper-mutation operator. Moreover, solutions of different optimization problems (unimodal, multi-modal, rotated multi-modal and composite problems) obtained by HLCSA are very stable. This is mainly caused by the diverse characteristics of the candidate Baldwinian learning strategies in the pool. Taking functions f_2,f_8 and f_{14} for example, these three problems are ill-conditioned and their global optima are not located at the symmetric centers of corresponding feasible solution spaces. The global optima of functions f_2, f_8 and f_{14} can be obtained at $[1, 1, \ldots, 1]$, $[420.96, 420.96, \ldots, 420.96]$ and $[420.96, 420.96, \ldots, 420.96]$, respectively. However, HLCSA can effectively adapt to different optimization stages and find the global optima of functions f_8 and f_{14}, and the approximate global optimum of function f_2.

Table 3 shows the results for CLONALG and HLCSA when $D = 30$, which indicate that HLCSA scales well in dealing with high-dimensional optimization problems. The best mean values are shown in bold face. The performance of HLCSA has no obvious degeneration for all the test problems except function f_{14} while CLONALG performs much worse. The local perturbation in CLONALG severely relies on the information of antibody itself, which leads to the lack of ability to jump out of local optima when the landscape of optimization problems are complex. However, the learning mechanism in HLCSA can help to pull the antibody out of local optima and effectively guide the evolutionary process towards the global optima.

4.3.2. Results of HLCSA and other EAs

Table 4 presents the mean values and standard deviations of the eight algorithms on the 16 test functions with $D = 10$, where the best results are shown in bold face. The best results among these eight algorithms are shown in boldface. As can be seen from the results, we observe that HLCSA surpasses all the other algorithms on functions f_2,f_4, f_{14} and f_{16} with great superiority. HLCSA can reach the global optima with probability 1 on functions f_4 and f_{14}. Moreover, HLCSA, BCSA, SaDE, and CLPSO can obtain the global minima 0 on functions f_5,f_6,f_7, and f_8; HLCSA, BCSA, SaDE, OXDE get the gloabl optimum of function f_{11} and HLCSA, SaDE, OXDE, GL-25 find the optimum on function f_{15}. HLCSA especially improves the results on functions f_2,f_4,f_{14} and f_{16}. Though HLCSA cannot get the best results on functions f_3,f_9,f_{10},f_{12} and f_{13}, it still get the comparable solutions with respect to the best results obtained by one of the other algorithms. HLCSA performs better on complex problems such as the ill-conditioned problems (functions f_2,f_8 and f_{14}) and the composite problems (functions f_{15} and f_{16}) while the other algorithms are easily trapped in local optima. HLCSA successfully avoids falling into deep local optima which are far away from the global optimum. OXDE shares similar properties with HLCSA, which makes the differential evolution and orthogonal learning work in parallel towards the excellent solution spaces; however, the orthogonal learning in HLCSA is performed after the Baldwinian learning stage in each generation, which is more like a local search operator to refine the solutions found in Baldwinian learning stage. OXDE can reach the optimal results on functions f_5,f_8,f_{11} and f_{15}. For function f_2, only OXDE can obtain a comparable result with that obtained by HLCSA. Without uniform initial population generated by orthogonal experimental design, OGA/Q cannot get promising results on most optimization problems. There may be two reasons accounting for this phenomenon: (1) the parameters used in the experiment are not the optimal values for running OGA/Q (e.g., the population size NP, the size of orthogonal array M, and the genetic algorithm related parameters P_c, P_m) and (2) the standard genetic algorithm framework, which employs the crossover and mutation as main operators to generate new schemes, is not effective enough for complex optimization problems.

Furthermore, the distance index

$$\mathrm{Dist}(D) = \frac{\log_{10}(f_{best}(D) - f_{opt}(D))}{D}, \tag{14}$$

Table 3
Results of traditional clonal selection algorithm and HLCSA when $D = 30$.

ALGs	Clonal selection algorithm				Hybrid learning clonal selection algorithm			
	Max	Min	Mean	Std	Max	Min	Mean	Std
f_1	1.4476e−001	2.6361e−002	6.4595e−002	2.9504e−002	1.1106e−064	1.9204e−068	**7.1289e−066**	2.0169e−065
f_2	3.6521e+001	2.2686e+001	2.7748e+001	2.1866e+000	3.0489e−014	4.7144e−020	**1.1617e−015**	5.5448e−015
f_3	7.7705e−001	1.1075e−001	3.1589e−001	1.8716e−001	2.6645e−015	2.6645e−015	**2.6645e−015**	0.0000e−000
f_4	2.4262e−001	5.9626e−002	1.6746e−001	4.3211e−002	0	0	**0**	0
f_5	7.8409e−001	4.9631e−001	6.3164e−001	7.5307e−002	0	0	**0**	0
f_6	4.4944e+001	2.9332e+001	3.6752e+001	4.6907e+000	0	0	**0**	0
f_7	2.8000e+001	1.8002e+001	2.3935e+001	2.5226e+000	0	0	**0**	0
f_8	1.8848e+003	1.1766e+003	1.6279e+003	1.7813e+002	0	0	**0**	0
f_9	4.5588e+000	3.1783e+000	3.8700e+000	3.2556e−001	3.5527e−015	3.5527e−015	**3.5527e−015**	0.0000e−000
f_{10}	9.3288e−001	6.1664e−001	8.3333e−001	7.5256e−002	0	0	**0**	0
f_{11}	3.9576e+001	2.6649e+001	3.3702e+001	2.7162e+000	0	0	**0**	0
f_{12}	2.9042e+002	2.3028e+002	2.5929e+002	1.4658e+001	3.7808e+001	1.2934e+001	**2.5471e+001**	6.7663e+000
f_{13}	3.0161e+002	2.0977e+002	2.5762e+002	2.1376e+001	8.1537e+001	1.6995e+001	**4.7609e+001**	1.5104e+001
f_{14}	9.3431e+003	7.9673e+003	8.7875e+003	3.2136e+002	2.4636e+003	2.1714e+002	**1.0663e+003**	5.2886e+002
f_{15}	6.3307e+001	2.8034e+001	4.4892e+001	9.5942e+000	0	0	**0**	0
f_{16}	4.9399e+001	3.3236e+001	3.9889e+001	4.3023e+000	5.3329e+000	9.0349e−001	**3.2212e+000**	1.0133e+000

Table 4
Results (mean ± std) of HLCSA and other state-of-the-art evolutionary algorithms when $D = 10$.

ALGs	f_1	f_2	f_3	f_4
HLCSA	4.2228e−053 ± 9.9892e−053	**3.9087e−028 ± 6.6496e−028**	2.5757e−015 ± 4.8648e−016	**0 ± 0**
BCSA	1.1694e−037 ± 1.3065e−037‡	1.8521e−001 ± 7.2808e−001‡	2.6645e−015 ± 0.0000e−000$^{\approx}$	1.4146e−001 ± 1.3744e−001‡
SaDE	1.4451e−176 ± 0.0000e−000†	2.0249e+000 ± 7.4832e−001‡	5.0330e−015 ± 1.1109e−015‡	1.8074e−003 ± 3.8251e−003‡
OXDE	4.5059e−056 ± 7.4966e−056†	1.0265e−026 ± 2.0786e−026‡	**2.0724e−015 ± 1.3467e−015**‡	9.9330e−004 ± 1.0673e−002‡
OGA/Q	5.3319e−001 ± 6.7874e−001‡	1.6235e+001 ± 2.1478e+001‡	4.8250e−001 ± 3.7514e−001‡	4.8189e−001 ± 2.2341e−001‡
CLPSO	1.8154e−041 ± 3.0360e−041‡	2.1490e+000 ± 1.2450e+000‡	3.9672e−015 ± 1.7413e−015‡	7.4577e−006 ± 2.1864e−005‡
CMA-ES	7.7614e−039 ± 2.2017e−039‡	5.3154e−001 ± 1.3783e+000‡	1.9403e+001 ± 3.0498e−001‡	8.9474e−003 ± 9.0610e−003‡
GL-25	**1.0771e−321 ± 0.0000e−000**†	2.0956e+000 ± 6.3579e−001‡	2.7830e−015 ± 1.4703e−015$^{\approx}$	1.2134e−002 ± 1.0199e−002‡
ALGs	f_5	f_6	f_7	f_8
HLCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
BCSA	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$
SaDE	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$
OXDE	**0 ± 0**$^{\approx}$	6.6331e−002 ± 2.5243e−001‡	5.6667e−001 ± 7.2793e−001‡	**0 ± 0**$^{\approx}$
OGA/Q	4.3288e−001 ± 1.3145e−001‡	3.1418e−001 ± 3.7811e−001‡	3.0388e−001 ± 1.9649e−001‡	1.2069e+000 ± 1.1358e+000‡
CLPSO	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$
CMA-ES	3.7346e−001 ± 6.5859e−001‡	7.6213e+001 ± 1.9296e+001‡	8.1200e+001 ± 2.8241e+001‡	1.7471e+003 ± 3.9521e+002‡
GL-25	7.3315e−007 ± 2.2405e−006‡	1.9633e+000 ± 1.1774e+000‡	5.6336e+000 ± 1.2724e+000‡	2.8952e+002 ± 1.9959e+002‡
ALGs	f_9	f_{10}	f_{11}	f_{12}
HLCSA	3.5527e−015 ± 0.0000e−000	2.8802e−002 ± 1.9129e−002	**0 ± 0**	4.2783e+000 ± 2.0095e+000
BCSA	3.5527e−015 ± 0.0000e−000$^{\approx}$	3.2715e−001 ± 1.7104e−001‡	**0 ± 0**$^{\approx}$	6.2881e+001 ± 1.6334e+001‡
SaDE	**9.4739e−016 ± 1.5979e−015**†	1.3704e−002 ± 1.6048e−002†	**0 ± 0**$^{\approx}$	3.9135e+000 ± 1.4295e+000†
OXDE	3.1974e−015 ± 1.0840e−015$^{\approx}$	4.9045e−001 ± 2.7411e−002‡	**0 ± 0**$^{\approx}$	3.7808e+000 ± 1.9094e+000†
OGA/Q	1.8265e+000 ± 8.2484e−001‡	7.4891e−001 ± 1.7034e−001‡	3.3346e+000 ± 9.2087e−001‡	1.1955e+001 ± 5.0220e+000‡
CLPSO	3.8606e−014 ± 5.8665e−014‡	2.8592e−002 ± 1.5307e−002$^{\approx}$	1.4403e−002 ± 1.2235e−002‡	4.1634e+000 ± 9.0362e−001†
CMA-ES	1.8759e+001 ± 3.5652e+000‡	1.0993e−002 ± 1.0707e−002†	5.1369e−001 ± 7.1191e−001‡	7.1669e+001 ± 2.3717e+001‡
GL-25	3.5527e−015 ± 0.0000e−000$^{\approx}$	**1.0545e−002 ± 1.0858e−002**†	2.1771e−004 ± 4.7010e−004‡	**3.3619e+000 ± 2.2861e+000**†
ALGs	f_{13}	f_{14}	f_{15}	f_{16}
HLCSA	4.2442e+000 ± 2.6469e+000	**0 ± 0**	**0 ± 0**	**4.6177e−001 ± 8.1247e−001**
BCSA	6.5935e+001 ± 1.0618e+001‡	2.6391e+002 ± 7.9682e+001‡	4.3387e−031 ± 6.5797e−031‡	8.7291e+000 ± 1.7669e+001‡
SaDE	3.9534e+000 ± 1.9500e+000†	2.0681e+002 ± 8.6302e+001‡	**0 ± 0**$^{\approx}$	1.8019e+001 ± 3.7308e+001‡
OXDE	3.0956e+000 ± 1.1245e+000†	1.5792e+001 ± 6.7669e+001‡	**0 ± 0**$^{\approx}$	1.0047e+001 ± 3.0498e+001‡
OGA/Q	1.1296e+001 ± 4.9557e+000‡	1.1708e+003 ± 4.4485e+002‡	6.0134e+001 ± 7.2387e+001‡	1.3740e+002 ± 1.5614e+002‡
CLPSO	**2.0254e+000 ± 1.0621e+000**†	3.1281e+002 ± 1.5723e+002‡	2.3104e−002 ± 6.5636e−002‡	6.0233e+000 ± 4.0698e+000‡
CMA-ES	7.8599e+001 ± 2.8068e+001‡	1.7538e+003 ± 6.7858e+002‡	1.2667e+002 ± 1.8370e+002‡	1.2878e+002 ± 2.2171e+002‡
GL-25	7.3657e+000 ± 2.2262e+000‡	5.2254e+002 ± 1.7963e+002‡	**0 ± 0**$^{\approx}$	9.0000e+001 ± 3.0513e+001‡

"$\dagger$" and "$\ddagger$" indicate the t value is significant at a 0.05 level of significance by two-tailed t-test. "$\dagger$", "$\ddagger$" and "$\approx$" denote the performance of corresponding algorithm is better than, worse than, and similar to that of HLCSA, respectively.

which was suggested in [55], is employed to measure the convergence speed of each algorithm. Fig. 2 presents the Dist(D) values in terms of the best fitness value of the median run of each algorithm for each problem ($D = 10$). We record the best solutions every 5000 function evaluations for each test problem with total function evaluations 100,000. Therefore, the interval for the horizontal coordinate is [1,20] and the vertical coordinate shows the value of Dist(D). It can be viewed from the convergence graphs, among these eight algorithms, HLCSA and SaDE have the comparatively best convergence characteristic for most test problems. SaDE employs different learning rules as well as adaptive control parameters according to the evolutionary history and thus it can adapt to different optimization stages and attain good convergence speed. However, SaDE has high computational cost on adapting the parameters F and CR. GL-25 shows good convergence speed on functions $f_1, f_3, f_9, f_{10}, f_{12}$ and f_{15}. CLPSO, which guides the evolutionary process by learning from all other particles' historical best

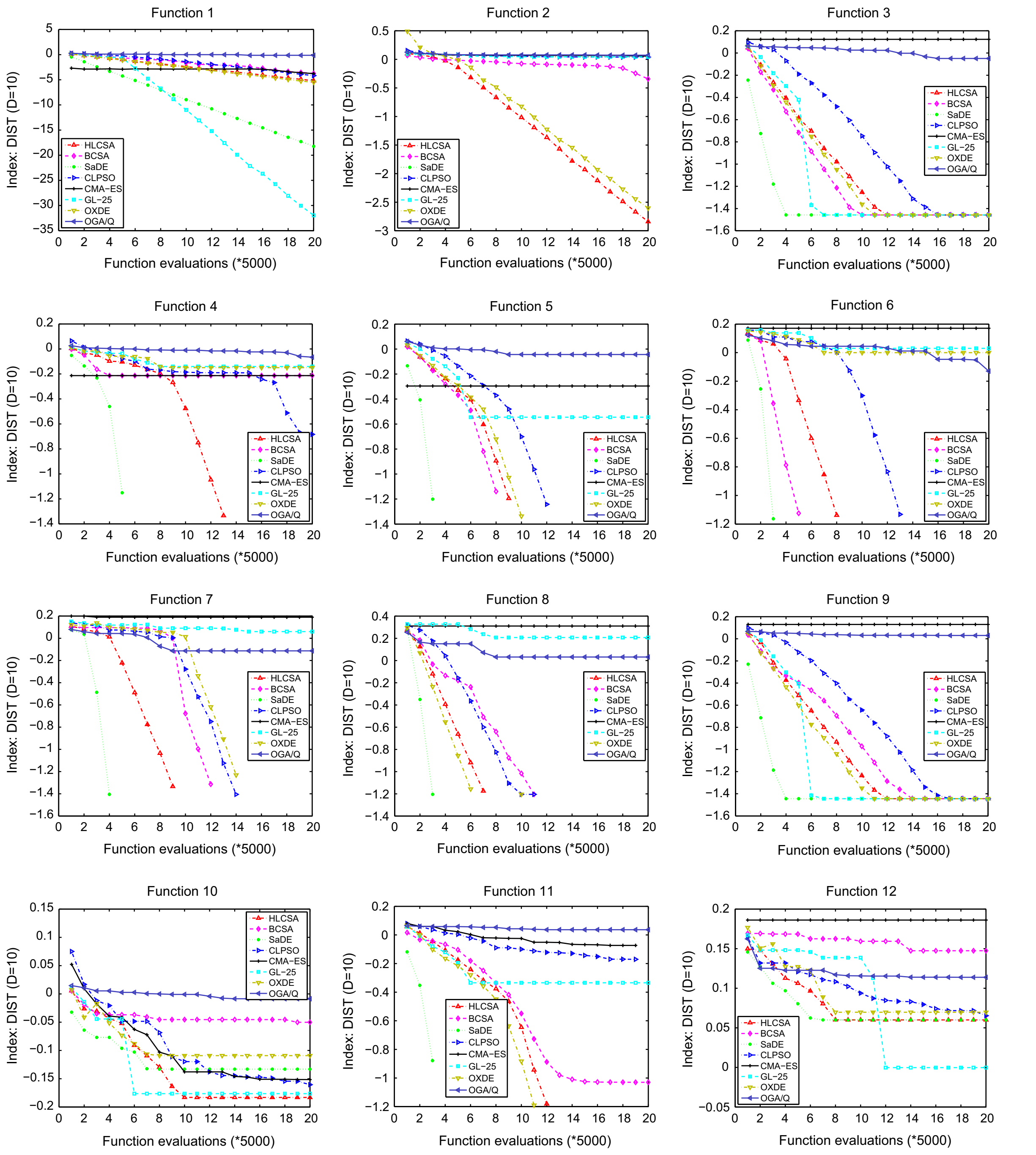

Fig. 2. The median dist(D) values of 10-D test functions.

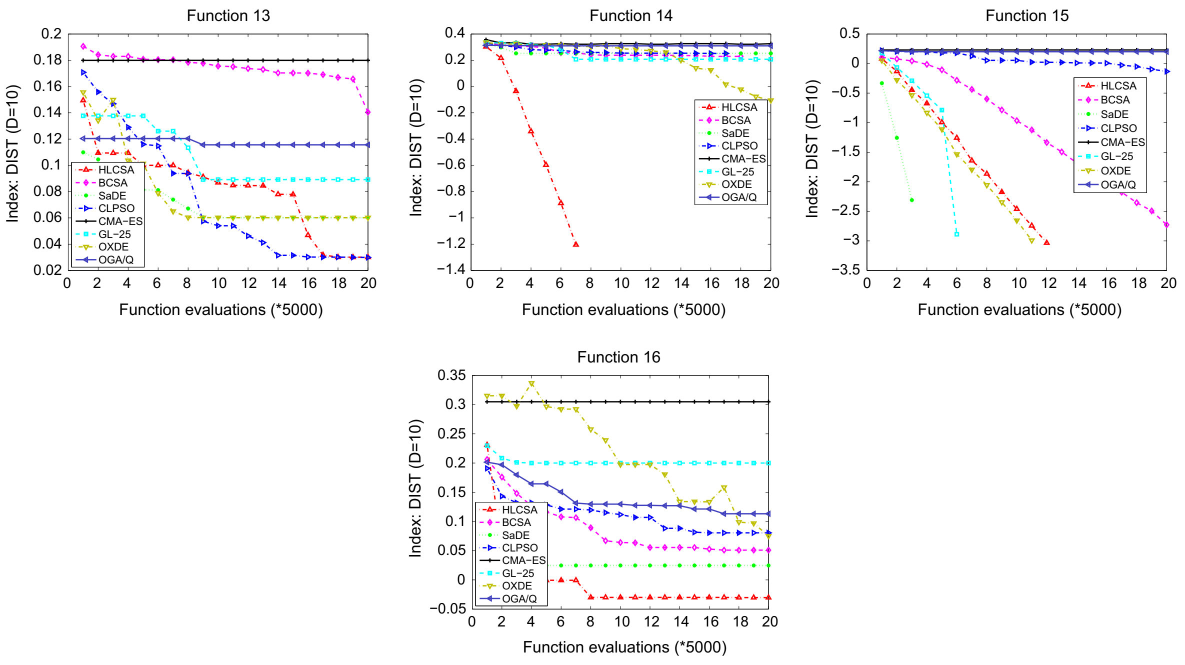

Fig. 2 (*continued*)

information, performs well on functions f_5, f_6, f_7, f_8 and f_{13}. HLCSA can be seen as an enhanced version of BCSA with two improvements: (1) using more Baldwinian learning rules to explore the feasible solution space and (2) incorporating the orthogonal learning strategy to refine the solutions; therefore, it consistently converges faster than BCSA. OXDE has similar convergence speed with HLCSA on functions f_2, f_3, f_5, f_{11} and f_{15}. Though OGA/Q employed the orthogonal crossover operator as well, it does not work on excellent parents; in other words, the subspace defined by parents is large or not promising and it is hard for orthogonal learning to search in such subspace effectively. Therefore, OGA/Q has slower convergence speed with respect to HLCSA.

The experiments conducted on 10-*D* problems are repeated on the 30-*D* problems and the results are presented in Table 5. Table 5 shows the scalability of HLCSA is pretty well when dealing with high dimensional optimization problems, where the best results are shown in bold face. It can consistently obtain good results when $D = 30$. Overall, the performance of HLCSA on 30-*D* problems are slightly better than those on 10-*D* problems. HLCSA can achieve the best performance on functions $f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}$ and f_{15}. Even HLCSA cannot find the best results on functions f_{12}, f_{13} and f_{16}, its obtained results are very close to the best results obtained by one of the other algorithms. For example, the best result on function f_{16} (2.5992e+000 ± 5.6332e−001) is found by GL-25 while a comparable result obtained by HLCSA is 3.2212e+000 ± 1.0133e+000. HLCSA greatly improves the results on functions f_2, f_9, f_{10}, f_{11} and f_{15}. It obtains comparatively good results on functions f_{12}, f_{13} and f_{14}. The convergence characteristics of HLCSA on the 30-*D* problems are similar with those when $D = 10$ and are not presented here.

4.4. Discussions

In this section, we will give some discussions on the proposed HLCSA from the following aspects:

- The sensitivity of parameter *s* which is the strength of Baldwinian learning.
- The performance comparison between OGA/Q and HLCSA in which orthogonal design are used to initialize a uniform population.
- The effect of the size of the orthogonal array.
- The performance comparison between HLCSA without orthogonal learning and the existing Baldwinian clonal selection algorithm (BCSA).
- The number of times of each the newly designed Baldwinian learning strategy in the candidate pool.

The strength of Baldwinian learning parameter *s* is approximated by a normal distribution with mean 0.5 and standard deviation 0.3, which follows the setting of the *scaling factor F* in [43]. For testing the sensitivity of *s*, we have tested another three variants of HLCSA in which *s* are fixed as 0.9, 0.5 and 0.3, respectively. We denote these three variants as HLCSA_S09, HLCSA_S05 and HLCSA_S03, respectively. The comparison results are presented in Table 6. From Table 6, we can see that HLCSA with *s* following the Gaussian distribution can obtain best result on most test instances. HLCSA has weaker results

Table 5
Results (mean ± std) of HLCSA and other state-of-the-art evolutionary algorithms when $D = 30$.

ALGs	f_1	f_2	f_3	f_4
HLCSA	7.1289e−066 ± 2.0169e−065	**1.1617e−015 ± 5.5448e−015**	**2.6645e−015 ± 0.0000e−000**	**0 ± 0**
BCSA	2.9665e−025 ± 8.4182e−025‡	1.9018e+001 ± 2.6925e+000‡	1.5614e−013 ± 3.9345e−013‡	**0 ± 0**$^{\approx}$
SaDE	9.1236e−150 ± 4.4538e−149†	2.1973e+001 ± 1.0132e+000‡	7.7383e−001 ± 6.0009e−001‡	1.1999e−002 ± 1.9462e−002‡
OXDE	4.8545e−059 ± 1.2064e−058‡	2.6577e−001 ± 1.0114e+000‡	2.6645e−001 ± 0.0000e−000‡	2.8730e−003 ± 5.6727e−003‡
OGA/Q	1.1781e+000 ± 7.6430e−001‡	9.2384e−001 ± 3.9658e−001‡	5.5912e−001 ± 1.8677e−001‡	8.1427e−001 ± 1.7221e−001‡
CLPSO	1.9761e−029 ± 1.5041e−029‡	1.7605e+001 ± 3.6364e+000‡	1.8415e−014 ± 3.0495e−015‡	1.1102e−016 ± 3.2467e−016‡
CMA-ES	6.6629e−029 ± 1.2542e−029‡	2.6577e−001 ± 1.0114e+000‡	1.9417e+001 ± 1.4026e−001‡	1.4782e−003 ± 4.2710e−003‡
GL-25	**5.3539e−228 ± 0.0000e−000**†	2.0832e+001 ± 8.6842e−001‡	8.4969e−014 ± 1.7664e−013‡	9.7959e−015 ± 3.2264e−014‡
ALGs	f_5	f_6	f_7	f_8
HLCSA	**0 ± 0**	**0 ± 0**	**0 ± 0**	**0 ± 0**
BCSA	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$
SaDE	9.5195e−002 ± 1.5798e−001‡	8.6230e−001 ± 8.9502e−001‡	6.3333e−001 ± 7.6489e−001‡	3.9479e+001 ± 6.4747e+001‡
OXDE	1.6214e−003 ± 6.5408e−003‡	9.4189e−000 ± 2.0859e−000‡	1.5100e+001 ± 2.9868e+000‡	3.9479e+000 ± 2.1624e+001‡
OGA/Q	1.2748e+000 ± 2.3090e−001‡	8.6041e−001 ± 4.0063e−001‡	1.6526e+000 ± 5.5564e−001‡	5.3729e+000 ± 6.5351e+000‡
CLPSO	**0 ± 0**$^{\approx}$	**0 ± 0**$^{\approx}$	8.7634e−015 ± 1.3333e−014‡	**0 ± 0**$^{\approx}$
CMA-ES	3.1331e+000 ± 2.2102e+000‡	2.3285e+002 ± 4.3316e+001‡	2.5993e+002 ± 3.8251e+001‡	5.3906e+003 ± 8.5755e+002‡
GL-25	7.1724e−004 ± 4.8027e−004‡	2.3030e+001 ± 8.4952e+000‡	3.9096e+001 ± 2.2071e+001‡	3.5030e+003 ± 6.8004e+000‡
ALGs	f_9	f_{10}	f_{11}	f_{12}
HLCSA	**3.5527e−015 ± 0.0000e−000**	**0 ± 0**	**0 ± 0**	2.5471e+001 ± 6.7663e+000
BCSA	1.3086e−013 ± 3.4341e−013‡	1.2273e−002 ± 2.2389e−002‡	1.8516e−014 ± 4.8488e−014‡	3.2115e+001 ± 9.0135e+000‡
SaDE	1.1708e+000 ± 6.5356e−001‡	1.3096e−002 ± 2.6787e−002‡	2.0504e+000 ± 9.0887e−001‡	2.5050e+001 ± 6.6415e+000$^{\approx}$
OXDE	**3.5527e−015 ± 0.0000e−000**$^{\approx}$	1.5612e−003 ± 3.2032e−003‡	1.4210e−001 ± 2.3765e−001‡	**1.6549e+001 ± 4.4609e+000**†
OGA/Q	2.5607e+000 ± 6.8983e−001‡	8.5935e−001 ± 1.5327e−001‡	9.6029e+000 ± 2.0326e+000‡	4.4368e+001 ± 1.4684e+001‡
CLPSO	1.4501e−007 ± 7.0645e−007‡	4.4152e−007 ± 7.4331e−007‡	1.7977e+000 ± 6.1941e−001‡	4.6287e+001 ± 5.7149e+000‡
CMA-ES	1.9486e+001 ± 1.5860e−001‡	6.5723e−004 ± 2.5834e−003‡	2.9295e+000 ± 1.9939e+000‡	2.2961e+002 ± 5.0531e+001‡
GL-25	1.1416e−013 ± 1.6841e−013‡	4.2040e−015 ± 5.5414e−015‡	5.6795e−003 ± 2.6720e−003‡	2.9464e+001 ± 2.2594e+001‡
ALGs	f_{13}	f_{14}	f_{15}	f_{16}
HLCSA	4.7609e+001 ± 1.5104e+001	1.0663e+003 ± 5.2886e+002	**0 ± 0**	3.2212e+000 ± 1.0133e+000
BCSA	2.6912e+001 ± 1.2152e+001†	2.6533e+003 ± 5.3339e+002‡	2.0259e−023 ± 6.3626e−023‡	1.8053e+001 ± 1.8282e+001‡
SaDE	2.2788e+001 ± 6.6879e+000†	2.4742e+003 ± 5.9013e+002‡	1.1833e−031 ± 2.1659e−031‡	1.0208e+001 ± 1.7851e+001‡
OXDE	**1.7959e+001 ± 5.1559e+000**†	**4.3428e+001 ± 8.8341e+001**†	3.3333e+000 ± 1.8257e+001‡	**2.5992e+000 ± 5.6332e−001**†
OGA/Q	4.0286e+001 ± 1.3281e+001†	4.0473e+003 ± 9.4893e+002‡	5.3477e+001 ± 9.3689e+001‡	2.3187e+001 ± 5.5504e+000‡
CLPSO	4.0333e+001 ± 7.5039e+000†	2.6321e+003 ± 3.3553e+002‡	8.2952e−005 ± 3.3295e−004‡	7.9983e+000 ± 1.6728e+000‡
CMA-ES	2.4410e+002 ± 4.9446e+001‡	6.7111e+003 ± 1.2362e+003‡	1.2000e+002 ± 1.6897e+002‡	7.6018e+001 ± 1.4964e+000‡
GL-25	9.6862e+001 ± 4.0914e+001‡	3.2335e+003 ± 5.9871e+002‡	2.7878e−028 ± 1.1207e−027‡	5.2187e+001 ± 2.1654e+001‡

Table 6
Sensitivity of HLCSA on the Baldwinian learning strength parameter s when $D = 10$.

FUNs	HLCSA Gaussian $s \sim N(0.5, 0.3)$	HLCSA_S09	HLCSA_S05	HLCSA_S03
		Fixed s = 0.9	Fixed s = 0.5	Fixed s = 0.3
f_1	4.22e−53 ± 9.98e−53	6.74e−46 ± 1.55e−45‡	1.62e−108 ± 8.23e−108†	4.00e−01 ± 8.87e−01‡
f_2	3.91e−28 ± 6.65e−28	3.54e−29 ± 6.22e−29†	6.44e+00 ± 1.26e+00‡	7.82e+00 ± 1.06e+00‡
f_3	2.58e−15 ± 4.86e−16	2.66e−15 ± 0.00e−00$^{\approx}$	1.78e−16 ± 1.66e−15†	8.18e−01 ± 8.59e−01‡
f_4	0 ± 0	2.49e−02 ± 1.70e−02‡	1.76e−02 ± 1.51e−02‡	7.76e−02 ± 4.27e−02‡
f_5	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$	3.87e−02 ± 1.23e−01‡
f_6	0 ± 0	4.64e−01 ± 7.27e−01‡	1.16e+00 ± 1.14e+00‡	1.51e+00 ± 9.48e−01‡
f_7	0 ± 0	1.39e+00 ± 1.23e+00‡	1.73e+00 ± 1.36e+00‡	2.57e+00 ± 1.27e+00‡
f_8	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$	5.57e+01 ± 8.66e+01‡
f_9	3.55e−15 ± 0.00e−00	3.32e−15 ± 9.01e−16$^{\approx}$	2.01e−15 ± 1.79e−15†	1.09e+00 ± 9.38e−01‡
f_{10}	2.88e−02 ± 1.91e−02	3.89e−02 ± 2.37e−02‡	2.92e−02 ± 3.03e−02$^{\approx}$	8.34e−02 ± 7.72e−02‡
f_{11}	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$	2.04e−01 ± 2.34e−01‡
f_{12}	4.28e+00 ± 2.01e+00	6.67e+00 ± 3.35e+00‡	4.94e+00 ± 2.87e+00‡	4.15e+00 ± 1.83e+00$^{\approx}$
f_{13}	4.24e+00 ± 2.65e+00	4.57e+00 ± 1.45e+00$^{\approx}$	7.07e+00 ± 2.35e+00‡	6.60e+00 ± 1.87e+00‡
f_{14}	0 ± 0	4.54e+01 ± 1.07e+00‡	1.49e+00 ± 1.69e+00‡	4.82e+02 ± 2.51e+02‡
f_{15}	0 ± 0	0 ± 0$^{\approx}$	1.43e+01 ± 3.63e+01‡	2.11e+01 ± 3.84e+01‡
f_{16}	4.62e−01 ± 8.12−01	1.34e−01 ± 4.11e−01†	2.30e+00 ± 2.48e+00‡	1.85e+01 ± 1.83e+01‡

on f_1,f_2,f_3 and f_9 with respect to its certain variants. However, the results of f_2, f_3 and f_9 obtained by HLCSA just have slight accuracies loss when compared with the best results. Therefore, we can conclude that it is better to use genus-random Baldwinian learning strength s instead of fixed values. The reason accounting for why this randomness can bring accuracy gain could be that it may introduce more variation to the search and thus strength the search ability.

Table 7
Performance comparison between OGA/Q and HLCSA in which the orthogonal design is used to initialize an uniform population.

FUNs	$D = 10$		$D = 30$	
	HLCSA	OGA/Q	HLCSA	OGA/Q
f_1	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_2	3.91e−28 ± 6.65e−28	6.55e+00 ± 2.56e+00	1.16e−15 ± 5.54e−15	9.24e−01 ± 3.97e−01
f_3	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_4	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_5	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_6	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_7	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_8	0 ± 0	2.52e+00 ± 2.93e+00	0 ± 0	5.97e+00 ± 3.61e+00
f_9	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_{10}	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_{11}	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_{12}	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_{13}	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_{14}	0 ± 0	2.29e+01 ± 8.54e+01	2.66e+ 00 ± 0.00e+00	6.50e+00 ± 3.81e+00
f_{15}	0 ± 0	2.13e+02 ± 1.89e+00	0 ± 0	4.02e+01 ± 1.04e+02
f_{16}	4.62e−01 ± 8.12e−01	1.44e+02 ± 1.95e+02	3.22e+00 ± 1.01e+00	9.00e+01 ± 0.00e+00

Table 8
Results (mean ± std) of HLCSA over 30 independent runs with varying sampling points of orthogonal learning when $D = 10$.

	$L_9(3^4)$	$L_{25}(5^6)$	$L_{49}(7^8)$
f_1	**4.22e−53 ± 9.98e−53**	7.22e−46 ± 1.57e−45	7.26e−38 ± 9.73e−38
f_2	**3.91e−28 ± 6.65e−28**	2.28e−23 ± 5.31e−23	8.66e−19 ± 1.32e−18
f_3	2.58e−15 ± 4.86e−16	**2.55e−15 ± 6.49e−16**	2.66e−15 ± 0.00e−00
f_4	**0 ± 0**	**0 ± 0**	1.50e−12 ± 8.20e−12
f_5	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_6	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_7	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_8	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_9	**3.55e−15 ± 0.00e−00**	3.83e−15 ± 6.49e−16	8.31e−14 ± 9.01e−15
f_{10}	2.88e−02 ± 1.91e−02	**2.03e−02 ± 1.22e−02**	2.58e−02 ± 1.79e−02
f_{11}	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{12}	4.28e+00 ± 2.01e+00	**2.65e+00 ± 1.23e+00**	3.42e+00 ± 1.63e+00
f_{13}	4.24e+00 ± 2.65e+00	4.09e+00 ± 1.29e+00	**3.04e+00 ± 1.03e+00**
f_{14}	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{15}	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{16}	4.62e−01 ± 8.12e−01	**3.53e−01 ± 7.47e−01**	3.87e−01 ± 6.08e−01

Table 9
Results (mean ± std) of HLCSA over 30 independent runs with varying sampling points of orthogonal learning when $D = 30$.

	$L_9(3^4)$	$L_{25}(5^6)$	$L_{49}(7^8)$
f_1	**7.13e−66 ± 2.02e−65**	2.05e−56 ± 6.84e−56	1.07e−46 ± 1.41e−46
f_2	**1.16e−15 ± 5.54e−15**	1.46e−11 ± 6.11e−11	1.33e−01 ± 7.28e−01
f_3	**2.66e−15 ± 0.00e−00**	2.66e−15 ± 0.00e−00	2.66e−15 ± 0.00e−00
f_4	**0 ± 0**	**0 ± 0**	3.28e−04 ± 1.79e−03
f_5	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_6	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_7	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_8	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_9	**3.55e−15 ± 0.00e−00**	3.55e−15 ± 0.00e−00	3.55e−15 ± 0.00e−00
f_{10}	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{11}	**0 ± 0**	**0 ± 0**	1.89e−01 ± 4.23e−01
f_{12}	2.55e+01 ± 6.77e+00	**2.44e+01 ± 7.73e+00**	2.58e+01 ± 8.02e+00
f_{13}	4.76e+01 ± 1.51e+01	2.83e+01 ± 6.50e+00	**2.79e+01 ± 7.17e+00**
f_{14}	1.07e+03 ± 5.29e+02	**3.78e+02 ± 3.14e+02**	1.44e+03 ± 4.61e+02
f_{15}	**0 ± 0**	**0 ± 0**	**0 ± 0**
f_{16}	**3.22e+00 ± 1.01e+00**	3.76e+00 ± 1.31e+00	3.88e+000 ± 9.69e−01

In Section 4.3.2, we run OGA/Q [46] without using orthogonal design to initialize the population. Here, we compare the performance between OGA/Q and HLCSA and use the orthogonal design to in both the algorithms to generate a uniform population. The other settings are exactly the same as described in Section 4.2. Table 7 presents the results obtained by these two algorithms. From Table 7, we can see that for both situations of $D = 10$ and $D = 30$, HLCSA can obtain better results on functions f_2,f_8,f_{14},f_{15} and f_{16}. The common characteristic of these five test instances is that the optimal solutions cannot be obtained in the center of the feasible solution space. Therefore, using orthogonal learning only is not sufficient for exploration.

In previous experiments, we use nine sample points of orthogonal learning for test instances over 10 and 30 dimensions. Though nine sample points are effective for generating promising results as described in Section 4.3.2, how the number of sample points of orthogonal experimental design affects the performance of HLCSA still need to be investigated. Therefore, we evaluate the performance of HLCSA by sampling 9($L_9(3^4)$), 25($L_{25}(5^6)$) and 49($L_{49}(7^8)$) points for test instances with 10 dimensions, and 9($L_9(3^4)$), 25($L_{25}(5^6)$) and 49($L_{49}(7^8)$) points for test instances with 30 dimensions. The experimental results are presented in Tables 8 and 9, where the best results are shown in bold face. We can find that $L_9(3^4)$ and $L_{25}(5^6)$ are more appropriate for both $D = 10$ and $D = 30$, which is consistent with the conclusion found in [44]. However, the trend is not obvious whether $L_{25}(5^6)$ is better than $L_9(3^4)$ when $D = 30$, which may be caused by the randomness of algorithm (e.g., using the Gaussian distributed Baldwinian learning strength s instead of a fixed value).

For evaluating the effectiveness of newly designed Baldwinian learning pool, we remove the orthogonal learning from the HLCSA and compared this modified HLCSA with single rule Baldwinian leaning method proposed in BCSA [25]. The experiments are conducted on benchmark test instances and the results are shown in Tables 10 and 11. Our experimental results demonstrate that this modified HLCSA can obtain better performance on all test instances except f_{13} when $D = 30$. This shows us that different optimization problems may need different trial vector generating rules and even for a single problem, different rules may be needed at different evolutionary stages. For example, we need to quickly locate the excellent solution space according to the guidance of the best individual in the population during the earlier evolutionary stages; and then we need to keep diversity to avoid premature during the last evolutionary stages (may be by introducing randomness). Furthermore, we include the results of HLCSA in Tables 10 and 11 to evaluate the validity of the orthogonal learning. Obviously, HLCSA can achieve better results than modified HLCSA without orthogonal learning on most test instances, which shows orthogonal learning is effective in refining the solutions. For the rotated multi-modal functions such as f_{10},f_{12} and f_{14}, the performance gap between HLCSA and modified HLCSA is obvious, which means that the mode 'global search with local search' is necessary for more complex optimization problems. In HLCSA, a number of function evaluations are consumed on calculating the fitness of solutions generated by orthogonal learning. Given the fixed number of function evaluations, the generations in HLCSA is fewer than that in modified HLCSA, which is the possible reason accounting for that the performance of HLCSA is worse than that of HLCSA without orthogonal learning in few test instances.

For making statistics to count the number of times of each newly designed strategy in the Baldwinian learning pool, we run HLCSA on each test instances for 30 independent times. The experimental results are shown in Fig. 3. As suggested by [43,42], multiple learning strategies are beneficial for generating better trial vectors. As shown in Fig. 3,, 'rand/1' learning rule is used most on all test instances. This accounts for the reason why Baldwinian clonal selection algorithm (BCSA) [25] used 'rand/1' as the only learning rule. However, the remaining learning rules are also important and used very often, especially on functions f_2, f_{10},f_{12} and f_{13} when $D = 10$.

Table 10
Results (mean ± std) of HLCSA, HLCSA without orthogonal learning and BCSA when $D = 10$.

Functions	HLCSA	Modified HLCSA	BCSA
f_1	4.22e−53 ± 9.99e−53	4.07e−140 ± 7.89e−140†	1.17e−37 ± 1.31e−37‡
f_2	3.91e−28 ± 6.65e−28	1.58e−20 ± 8.65e−20‡	1.85e−01 ± 7.28e−01‡
f_3	2.58e−15 ± 4.86e−16	0 ± 0†	2.66e−15 ± 1.11e−15‡
f_4	0 ± 0	5.09e−03 ± 8.40e−03‡	1.41e−01 ± 1.37e−01‡
f_5	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_6	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_7	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_8	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_9	3.55e−15 ± 0.00e−00	0 ± 0†	3.55e−15 ± 0.00e−00‡
f_{10}	2.88−02 ± 1.91e−02	1.67e−02 ± 1.55e−02†	3.27e−01 ± 1.71e−01‡
f_{11}	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_{12}	4.27e+00 ± 2.01e+00	5.07e+00 ± 2.30e+00‡	6.29e+01 ± 1.63e+01‡
f_{13}	4.24e+00 ± 2.65e+00	5.47e+00 ± 1.31e+00‡	6.59e+01 ± 1.06e+01‡
f_{14}	0 ± 0	1.31e+02 ± 1.79e+02‡	2.64e+02 ± 7.97e+01‡
f_{15}	0 ± 0	0 ± 0$^{\approx}$	4.34e−31 ± 6.58e−31‡
f_{16}	4.62e−01 ± 8.12e−01	9.12e−02 ± 3.50e−01†	8.73e+00 ± 1.77e+01‡

"†", "‡" and "≈" in the third column denote the performance of modified HLCSA is better than, worse than, and similar to that of HLCSA, respectively.
"†", "‡" and "≈" in the fourth column denote the performance of BCSA is better than, worse than, and similar to that of modified HLCSA, respectively.

Table 11
Results (mean ± std) of HLCSA, HLCSA without orthogonal learning and BCSA when D = 30.

Functions	HLCSA	Modified HLCSA	BCSA
f_1	7.13e−66 ± 2.02e−65	1.33e−102 ± 2.39e−102†	2.97e−25 ± 8.42e−25‡
f_2	1.16e−15 ± 5.54e−15	6.55e−10 ± 2.30e−09‡	1.90e+01 ± 2.69e+00‡
f_3	2.66e−15 ± 0.00e−00	5.98e−14 ± 9.01e−14‡	1.56e−13 ± 3.93e−13‡
f_4	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_5	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_6	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_7	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_8	0 ± 0	0 ± 0$^{\approx}$	0 ± 0$^{\approx}$
f_9	3.55e−15 ± 0.00e−00	0 ± 0†	1.31e−13 ± 3.43e−13‡
f_{10}	0 ± 0	3.20e−03 ± 6.04e−03‡	1.22e−02 ± 2.24e−02‡
f_{11}	0 ± 0	0 ± 0$^{\approx}$	1.85e−14 ± 4.85e−14‡
f_{12}	2.55e+01 ± 6.77e+00	2.74e+01 ± 8.08e+00‡	3.21e+01 ± 9.01e+00‡
f_{13}	4.76e+01 ± 1.51e+01	2.78e+01 ± 8.54e+00†	2.69e+001 ± 1.21e+01†
f_{14}	1.06e+03 ± 5.29e+02	1.96e+03 ± 7.44e+02‡	2.65e+003 ± 5.33e+02‡
f_{15}	0 ± 0	0 ± 0$^{\approx}$	2.02e−023 ± 6.36e−23‡
f_{16}	3.22e+00 ± 1.01e+00	4.05e+00 ± 1.29e+00‡	1.80e+001 ± 1.83e+01‡

"†", "‡" and "≈" in the third column denote the performance of modified HLCSA is better than, worse than, and similar to that of HLCSA, respectively.
"†", "‡" and "≈" in the fourth column denote the performance of BCSA is better than, worse than, and similar to that of modified HLCSA, respectively.

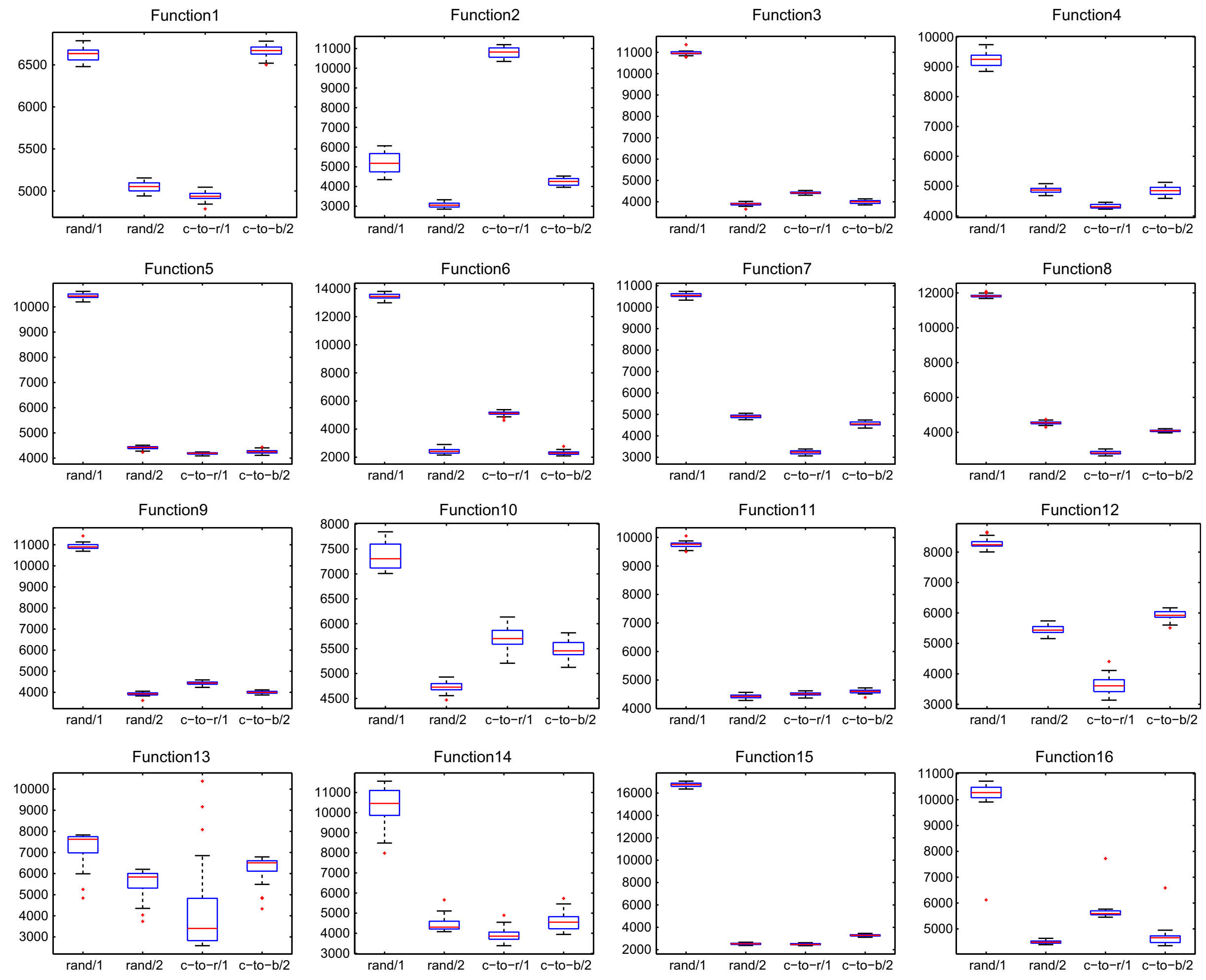

Fig. 3. The statistical times of each Baldwinian learning strategy used in optimizing $f_1 \sim f_8$. "rand/1","rand/2","c-to-r/1", "c-to-b/2" are respectively the abbreviations of "BL/rand/1", "BL/rand/2", "BL/current-to-rand/1" and "BL/current-to-best/2" defined in Section 3.1. The statistical times of each Baldwinian learning strategy used in optimizing $f_9 \sim f_{16}$. "rand/1","rand/2","c-to-r/1", "c-to-b/2" are respectively the abbreviations of "BL/rand/1", "BL/rand/2", "BL/current-to-rand/1" and "BL/current-to-best/2" defined in Section 3.1.

5. Conclusions and future work

The paper presented an enhanced clonal selection algorithm based on the idea that learning strategy could guide the evolutionary process based on hyper-mutation in CSA. Within the proposed framework, the hybrid learning clonal selection algorithm (HLCSA), two learning mechanisms were introduced: the Baldwinian learning and orthogonal learning. In the Baldwinian learning stage, four widely used strategies form the candidate pool which provides more flexibility to diverse optimization problems; in the orthogonal learning stage, the hyper-rectangle space defined by the selected antibody and its best Baldwinian learning vector is probed systematically. The experimental results on 16 test problems demonstrated that the proposed HLCSA has excellent performance for a variety of optimization problems.

In HLCSA, after Baldwinian learning, the best antibody in the four Baldwinian learning vectors originating from the same antibody are selected to form a new population $\mathbf{Z}(t)$; then the orthogonal learning strategy is employed to search in the space defined by $\vec{a}_k(t)$ and $\vec{z}_k(t)$. Here, the similarity between $\vec{a}_k(t)$ and $\vec{z}_k(t)$ is neglected. However, it is important to produce a small but representative set of points as the potential offspring according to the similarity of two parents. Thus, the most promising way is to adaptively adjust the number of orthogonal array's factors and location for dividing the parents into several sub-vectors. Doing this may also alleviate the burden of evaluating the offspring generated by orthogonal design in each generation. Our future work will focus on two aspects: (1) Investigate the adaptive orthogonal learning mechanism based on the similarity between parents, which will use different orthogonal arrays for different parent pairs and thus partition the space (defined by parent pair) into different number of subspaces. (2) Both clonal selection algorithm and orthogonal design are widely used in solving multi-objective optimization problem and demonstrate excellent performance [56,11,12]. The hybrid learning clonal selection algorithm (HLCSA) is proposed for global optimization problems in this paper. Therefore, extending the proposed algorithm to tackle multi-objective optimization problems (MOPs) is the other future work.

Acknowledgments

The authors would like to thank anonymous reviewers for their helpful comments, Prof. Ponnuthurai Nagaratnam Suganthan for providing the source codes of 'SaDE' and 'CLPSO', and Prof. Yong Wang for providing the source code of 'OXDE'. This work was supported partially by the National Natural Science Foundation of China (Grant No. 61272248), the National Basic Research Program of China (Grant No. 2013CB329401), the Science and Technology Commission of Shanghai Municipality (Grant No. 13511500200) and the European Union Seventh Framework Program (Grant No. 247619). The first author was supported by China Scholarship Council (Grant No. 201206230012) and this work was mainly done when he was visiting the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor.

References

[1] L.N. De Castro, F.J. Von Zuben, Learning and optimization using the clonal selection principle, IEEE Trans. Evolution. Comput. 6 (3) (2002) 239–251.
[2] B.H. Ulutas, S. Kulturel-Konak, A review of clonal selection algorithm and its applications, Artif. Intell. Rev. 36 (2) (2011) 117–138.
[3] D. Dasgupta, S. Yu, F. Nino, Recent advances in artificial immune systems: models and applications, Appl. Soft Comput. 11 (2) (2011) 1574–1587.
[4] M. Villalobos-Arias, C.A.C. Coello, O. Hernández-Lerma, Convergence analysis of a multiobjective artificial immune system algorithm, in: Proceedings of International Conference on Artificial Immune Systems, 2004, pp. 226–235.
[5] A. Hone, J. Kelsey, Optima, extrema, and artificial immune systems, in: Proceedings of International Conference on Artificial Immune Systems, 2004, pp. 80–90.
[6] J. Timmis, A. Hone, T. Stibor, E. Clark, Theoretical advances in artificial immune systems, Theoret. Comp. Sci. 403 (1) (2008) 11–32.
[7] T. Jansen, C. Zarges, Analyzing different variants of immune inspired somatic contiguous hypermutations, Theoret. Comp. Sci. 412 (6) (2011) 517–533.
[8] N. Khilwani, A. Prakash, R. Shankar, M. Tiwari, Fast clonal algorithm, Eng. Appl. Artif. Intell. 21 (1) (2008) 106–128.
[9] L. Jiao, Y. Li, M. Gong, X. Zhang, Quantum-inspired immune clonal algorithm for global optimization, IEEE Trans. Syst., Man, Cybernet., Part B: Cybernet. 38 (5) (2008) 1234–1253.
[10] M. Gong, L. Jiao, F. Liu, W. Ma, Immune algorithm with orthogonal design based initialization, cloning, and selection for global optimization, Knowl. Inform. Syst. 25 (3) (2010) 523–549.
[11] M. Gong, L. Jiao, H. Du, L. Bo, Multiobjective immune algorithm with nondominated neighbor-based selection, Evolution. Comput. 16 (2) (2008) 225–255.
[12] R. Shang, L. Jiao, F. Liu, W. Ma, A Novel Immune Clonal Algorithm for MO Problems, IEEE Trans. Evolution. Comput. 16 (1) (2012) 35–50.
[13] M. Gong, L. Zhang, L. Jiao, W. Ma, Differential immune clonal selection algorithm, in: Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems, 2007, pp. 666–669.
[14] X. Fu, A. Li, L. Wang, C. Ji, Short-term scheduling of cascade reservoirs using an immune algorithm-based particle swarm optimization, Comp. Math. Appl. 62 (6) (2011) 2463–2471.
[15] S. Afaneh, R.A. Zitar, A. Al-Hamami, Virus detection using clonal selection algorithm with genetic algorithm (VDC algorithm), Appl. Soft Comput. 13 (1) (2013) 239–246.
[16] R. Liu, L. Jiao, X. Zhang, Y. Li, Gene transposon based clone selection algorithm for automatic clustering, Inform. Sci. 204 (2012) 1–22.
[17] W. Ma, Y. Huang, C. Li, J. Liu, Image segmentation based on a hybrid immune memetic algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, 2012, pp. 1–8.
[18] L. Batista, F.G. Guimarães, J.A. Ramírez, A distributed clonal selection algorithm for optimization in electromagnetics, IEEE Trans. Magnet. 45 (3) (2009) 1598–1601.
[19] G.-Y. Xi, J.-P. Yue, B.-X. Zhou, P. Tang, Application of an artificial immune algorithm on a statistical model of dam displacement, Comp. Math. Appl. 62 (10) (2011) 3980–3986.
[20] Y. Zhong, L. Zhang, B. Huang, P. Li, An unsupervised artificial immune classifier for multi/hyperspectral remote sensing imagery, IEEE Trans. Geosci. Rem. Sens. 44 (2) (2006) 420–431.
[21] L. Zhang, Y. Zhong, B. Huang, J. Gong, P. Li, Dimensionality reduction based on clonal selection for hyperspectral imagery, IEEE Trans. Geosci. Rem. Sens. 45 (12) (2007) 4172–4186.

[22] Y. Zhong, L. Zhang, An adaptive artificial immune network for supervised classification of multi-/hyperspectral remote sensing imagery, IEEE Trans. Geosci. Rem. Sens. 50 (3) (2012) 894–909.
[23] Y. Peng, B.-L. Lu, Immune clonal algorithm based feature selection for epileptic EEG signal classification, in: Proceedings of IEEE International Conference on Information Science, Signal Processing and their Applications, 2012, pp. 848–853.
[24] J. Zheng, Y. Chen, W. Zhang, A Survey of artificial immune applications, Artif. Intell. Rev. 34 (1) (2010) 19–34.
[25] M. Gong, L. Jiao, L. Zhang, Baldwinian learning in clonal selection algorithm for optimization, Inform. Sci. 180 (8) (2010) 1218–1236.
[26] N.Q. Huy, O.Y. Soon, L.M. Hiot, N. Krasnogor, Adaptive cellular memetic algorithms, Evolution. Comput. 17 (2) (2009) 231–256.
[27] N. Ferrante, C. Carlos, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evolution. Comput. 2 (1) (2012) 1–14.
[28] M. Gong, L. Jiao, X. Zhang, A population-based artificial immune system for numerical optimization, Neurocomputing 72 (1) (2008) 149–161.
[29] K.W. Ku, M. Mak, Exploring the effects of Lamarckian and Baldwinian learning in evolving recurrent neural networks, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1997, pp. 617–621.
[30] G.E. Hinton, S.J. Nowlan, How learning can guide evolution, Comp. Syst. 1 (1) (1987) 495–502.
[31] K.W. Ku, M.-W. Mak, Empirical analysis of the factors that affect the Baldwin effect, in: Proceedings of International Conference on Parallel Problem Solving from Nature, 1998, pp. 481–490.
[32] Q. Yuan, F. Qian, W. Du, A hybrid genetic algorithm with the Baldwin effect, Inform. Sci. 180 (5) (2010) 640–652.
[33] M. Zhang, A novel sexual adaptive genetic algorithm based on Baldwin effect for global optimization, Int. J. Intell. Comput. Cybernet. 4 (2) (2011) 207–227.
[34] L. Zhang, M. Gong, L. Jiao, J. Yang, Improved Clonal Selection Algorithm based on Baldwinian learning, in: Proceedings of IEEE Congress on Evolutionary Computation, 2008, pp. 519–526.
[35] Y. Qi, F. Liu, M. Liu, M. Gong, L. Jiao, Multi-objective immune algorithm with Baldwinian learning, Appl. Soft Comput. 12 (8) (2012) 2654–2674.
[36] C. Zhang, J. Chen, B. Xin, Distributed memetic differential evolution with the synergy of Lamarckian and Baldwinian learning, Appl. Soft Comput. 13 (5) (2013) 2947–2959.
[37] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optimiz. 11 (4) (1997) 341–359.
[38] D. Federici, Culture and the Baldwin effect, in: Proceedings of European Conference on Artificial Life, 2003, pp. 309–318.
[39] P. Turney, How to shift bias: lessons from the Baldwin effect, Evolution. Comput. 4 (3) (1997) 271–295.
[40] E. Mezura-Montes, J. Velázquez-Reyes, C.A. Coello Coello, A comparative study of differential evolution variants for global optimization, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 485–492.
[41] R. Gämperle, S.D. Müller, P. Koumoutsakos, A parameter study for differential evolution, advances in Intelligent Systems, Fuzzy Syst., Evolution. Comput. 10 (2002) 293–298.
[42] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evolution. Comput. 15 (1) (2011) 55–66.
[43] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evolution. Comput. 13 (2) (2009) 398–417.
[44] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inform. Sci. 185 (1) (2012) 153–177.
[45] Q. Zhang, Y.-W. Leung, An orthogonal genetic algorithm for multimedia multicast routing, IEEE Trans. Evolution. Comput. 3 (1) (1999) 53–62.
[46] Y.-W. Leung, Y. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, IEEE Trans. Evolution. Comput. 5 (1) (2001) 41–53.
[47] W. Gong, Z. Cai, L. Jiang, Enhancing the performance of differential evolution using orthogonal design method, Appl. Math. Comput. 206 (1) (2008) 56–69.
[48] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, S.-J. Ho, OPSO: Orthogonal particle swarm optimization and its application to task assignment problems, IEEE Trans. Syst., Man Cybernet., Part A: Syst. Hum. 38 (2) (2008) 288–298.
[49] Z.-H. Zhan, J. Zhang, Y. Li, Y.-H. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evolution. Comput. 15 (6) (2011) 832–847.
[50] Y. Wang, H. Liu, Z. Cai, Y. Zhou, An orthogonal design based constrained evolutionary optimization algorithm, Eng. Optimiz. 39 (6) (2007) 715–736.
[51] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evolution. Comput. 10 (3) (2006) 281–295.
[52] J. Liang, P. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: Proceedings of IEEE Swarm Intelligence Symposium, 2005, pp. 68–75.
[53] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolution. Comput. 9 (2) (2001) 159–195.
[54] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, A.M. Sánchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, Euro. J. Operat. Res. 185 (3) (2008) 1088–1113.
[55] Y. Peng, B.-L. Lu, A hierarchical particle swarm optimizer with latin sampling based memetic algorithm for numerical optimization, Appl. Soft Comput. 13 (5) (2013) 2823–2836.
[56] S.-Y. Zeng, G. Chen, L. Zheng, H. Shi, H. de Garis, L. Ding, L. Kang, A dynamic multi-objective evolutionary algorithm based on an orthogonal design, in: Proceedings of IEEE Congress on Evolutionary Computation, 2006, pp. 573–580.

Yong Peng received the B.S. degree from Hefei New Star Research Institute of Applied Technology, the M.S. degree from Graduate University of Chinese Academy of Sciences, both in computer science, in 2006 and 2010, respectively. Now he is working towards his Ph.D. degree at Shanghai Jiao Tong University. He was awarded by the Presidential Scholarship, Chinese Academy of Sciences in 2009 and the National Scholarship for Graduate Students, Ministry of Education in 2012. His research interests are evolutionary computation, machine learning and pattern recognition.

Bao-Liang Lu received his B.S. degree from Qingdao University of Science and Technology, China in 1982, the M.S. degree from Northwestern Polytechnical University, China in 1989 and the Ph.D. degree from Kyoto University, Japan, in 1994. From 1982 to 1986, he was with the Qingdao University of Science and Technology. From April 1994 to March 1999, he was a Frontier Researcher at the Bio-Mimetic Control Research Center, the Institute of Physical and Chemical Research (RIKEN), Japan. From April 1999 to August 2002, he was a Research Scientist at the RIKEN Brain Science Institute. Since August 2002, he has been a full Professor at the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include brain-like computing, neural networks, machine learning, pattern recognition, and brain–computer interface. He was the past President of the Asia Pacific Neural Network Assembly (APNNA) and the general Chair of ICONIP2011. He serves on the editorial board of *Neural Networks Journal* (Elsevier). He is a governing board member of APNNA and a senior member of IEEE.