Applied Soft Computing 13 (2013) 2947–2959

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Distributed memetic differential evolution with the synergy of Lamarckian and Baldwinian learning

Chunmei Zhang[a,b], Jie Chen[a,c], Bin Xin[a,d,*]

[a] School of Automation, Beijing Institute of Technology, Beijing 100081, China
[b] School of Electronic Information Engineering, Taiyuan University of Science and Technology, Taiyuan 030024, China
[c] Key Laboratory of Complex System Intelligent Control and Decision, Ministry of Education, Beijing 100081, China
[d] Decision and Cognitive Sciences Research Centre, Manchester Business School, The University of Manchester, Manchester M15 6PB, UK

ARTICLE INFO

Article history:
Received 8 October 2011
Received in revised form 4 February 2012
Accepted 28 February 2012
Available online 16 March 2012

Keywords:
Distributed differential evolution
Memetic algorithm
Lamarckian learning
Baldwinian learning
Hooke–Jeeves algorithm

ABSTRACT

As a population-based optimizer, the differential evolution (DE) algorithm has a very good reputation for its competence in global search and numerical robustness. In view of the fact that each member of the population is evaluated individually, DE can be easily parallelized in a distributed way. This paper proposes a novel distributed memetic differential evolution algorithm which integrates Lamarckian learning and Baldwinian learning. In the proposed algorithm, the whole population is divided into several subpopulations according to the von Neumann topology. In order to achieve a better tradeoff between exploration and exploitation, the differential evolution as an evolutionary frame is assisted by the Hooke–Jeeves algorithm which has powerful local search ability. We incorporate the Lamarckian learning and Baldwinian learning by analyzing their characteristics in the process of migration among subpopulations as well as in the hybridization of DE and Hooke–Jeeves local search. The proposed algorithm was run on a set of classic benchmark functions and compared with several state-of-the-art distributed DE schemes. Numerical results show that the proposed algorithm has excellent performance in terms of solution quality and convergence speed for all test problems given in this study.

1. Introduction

The differential evolution (DE) is a stochastic, population-based global search and optimization method [1]. It uses the difference of solutions to create new candidate solutions and one-to-one spawning competition scheme to select new individuals greedily. These attractive characteristics make DE retain the knowledge of good solutions in the current population. DE, just like the particle swarm optimization (PSO) and genetic algorithm (GA) which have been implemented to various domains [2,3], has been proven to have a good performance on many real-world problems [4]. DE is good at exploring the search space and locating the region of global optima, but it is slow at fine-tuning the solution [5]. Some modifications on the basic DE can improve its performance, which can be grouped into two categories. One depends on the modifications of DE itself including its parameters, operator, and population structure [6,7]; and the other focuses on hybridizing DE with different optimizers such as additional local searchers and other population-based metaheuristics (e.g. particle swarm optimizer) [8–10].

In the case of modifying the population structure of DE, a popular way is dividing the whole population into multiple subpopulations which evolve independently and exchange information mutually. An important motivation behind the multiple-population strategy is to maintain population diversity and achieve the parallel search of the solution space. DE can be easily parallelized due to the fact that each member of the population is evaluated individually. Tasoulis et al. [11] explored how differential evolution can be parallelized by using a unidirectional ring topology and proposed an algorithm, namely parallel differential evolution (PDE), to improve both the speed and the performance of the method. Besides, a lot of research investigated the migration policy of parallel differential evolution, including migration schemes, migration frequencies, the number and size of subpopulations, and so on. Specially, for solving the learning issue in fuzzy neural inference system, Singh et al. [12] studied the parameterization of a parallel distributed DE in detail and discussed the influence of different interval and sizes of migration, as well as different number of islands. Apolloni et al. [13] designed a modified version of the PDE in a generic way, namely island based distributed differential evolution (IBDDE). A set of five parameters was integrated to elaborate on the main principles of the migration. A distributed

* Corresponding author at: School of Automation, Beijing Institute of Technology, Beijing 100081, China. Tel.: +86 1068912463; fax: +861068918233.
E-mail addresses: zcm10606@163.com (C. Zhang), chenjie@bit.edu.cn (J. Chen), brucebin@bit.edu.cn (B. Xin).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2012.02.028

version of the differential evolution (DDE) was coupled with affine transformation and mutual information maximization to perform the registration of remotely sensed images in [14]. This algorithm differs from PDE and IBDDE by the topology it adopts. Instead of a unidirectional ring, DDE uses a locally connected topology named torus topology. Based on comparative experiments, Falco et al. indicated that DDE is very promising to achieve better performance. Additionally, Izzo et al. [15] presented a heterogeneous asynchronous island model for DE. The results confirmed that such a model could improve the reliability and speed of the algorithm and find significantly better solutions. Recently, Matthieu et al. [16] designed an adaptive mechanism for the scale factor in distributed differential evolution schemes, called "F" adaptive control parallel differential evolution (FACPDE). The empirical results in [14] showed that the employment of multiple scale factors can greatly improve the performance of the distributed algorithm.

In the context of search and optimization, it is worth noting that the key design issue of evolutionary algorithms lies in the successful promotion of tradeoff between exploration and exploitation [17]. Krasnogor and Smith [18] pointed out that the population-based intelligent methods combining local search methods, called memetic algorithms (MAs) which were originally proposed by Moscato and Norman for the traveling salesman problem [19], can lead to a better tradeoff between exploration and exploitation and thereby improved performance. Now, the term MA is widely employed as a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem solving [20]. There are two mechanisms on how learning influences evolution. One is the Lamarckian learning (L-learning) in which the characteristics of phenotype and genotype acquired by an organism during its lifetime is transferred and can be passed on to the organism's offspring directly. Another is the Baldwinian learning (B-learning), different from the L-learning that the acquired genotype traits are not inherited to its offspring directly, but adaptive learning can guide the course of evolution indirectly in a way that learning alters the shape of search space and thereby provides good evolutionary paths towards individuals. The experimental results of [20] show that the L-learning in general has a higher performance than the B-learning, and most MAs employ the L-learning mechanism to achieve the combination of global search and local search. At the same time, Nguyen et al. [20] also pointed out that the L-learning cannot distinguish individuals effectively, easily leading to stagnation. Though comparatively time-consuming, it is easier to obtain global optima by the B-learning.

In view of above, this paper proposes a novel distributed memetic differential evolution (abbr., DMDE) which integrates the L-learning and the B-learning. In the proposed algorithm, the initial population is distributed over multiple subpopulations according to the von Neumann topology. All subpopulations interact by migrating their respective best individual to neighboring subpopulations and replacing the worst ones partly or entirely. In the evolutionary loop, DE is in charge of global search and the Hook–Jeeves algorithm assists DE to achieve local improvement. In order to balance exploration and exploitation, we incorporate the L-learning and the B-learning by analyzing their characteristics of individual learning. Then we achieve the cooperation in two aspects: one is the migration among subpopulations and the other is the hybridization of DE and the Hook–Jeeves algorithm.

The presented algorithm was run on a set of benchmark problems and compared with several distributed DE schemes. Numerical results show that the proposed DMDE makes a good tradeoff between exploration and exploitation and performs better than three state-of-the-art distributed DE algorithms for tackling both unimodal and multimodal problems.

The remainder of this paper is organized as follows. In Section 2, a basic DE algorithm is briefly introduced. Section 3 proposes the new distributed DE—DMDE. Section 4 analyses the computational complexity of the proposed scheme and presents the experimental results on benchmark functions. This section gives computational complexity and performance comparisons with several state-of-the-art distributed DE schemes as well as a discussion of the obtained results. Conclusion is summarized in Section 5.

2. Basic differential evolution

Differential evolution (DE) is a population-based metaheuristic. DE/rand/1/bin is one of the classic and most successful DE variants [21]. It generates new solution vectors by adding the weighted difference of two randomly selected population members to the third member, and forms the final trial vector with binomial crossover. In addition, DE employs a one-to-one spawning logic which allows replacement of an individual only if the offspring has better fitness value than its corresponding parent.

DE evolves NP D-dimensional individual vectors $\mathbf{x}_{i,g}$, $i = 1, 2, \ldots, NP$, where g denotes the current generation, and NP is the population size. The initial population is randomly generated within the whole search space. After initialization, DE performs in sequence three vector operations: differential mutation, crossover and selection.

A number of variations to the classic DE have been developed. Different DE strategies differ in the way that the base vector is selected, the number of difference vectors used, and the way that crossover points are determined. In order to characterize these variations, a general notation is adopted, namely DE/x/y/z [21]. x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z is the type of crossover being used. A brief introduction on the well-known DE variant DE/rand/1/bin is given in the following.

2.1. Differential mutation

For each target vector $\mathbf{x}_{i,g}$, $\{i = 1, 2, \ldots, NP\}$, the corresponding mutant vector $\mathbf{v}$ is generated as follows:

$$\mathbf{v} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}), \quad r_0, r_1, r_2 \in \{1, 2, \ldots, NP\}. \tag{1}$$

where $\mathbf{x}_{r0,g}$ is a base vector and the indexes satisfy $r_0 \neq r_1 \neq r_2 \neq i$. It is obvious that at least four individuals are needed to implement the above mutation operation, implying that $NP \geq 4$. The scaling factor F lies within the range (0, 2), usually less than 1 [22].

2.2. Crossover

After mutation, the target vector (individual) is mixed with the mutant vector, and the following crossover operation is used to form the final trial vector:

$$\mathbf{u}_{i,g,j} = \begin{cases} \mathbf{v}_{i,g,j}, & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ \mathbf{x}_{i,g,j}, & \text{otherwise} \end{cases}, \tag{2}$$

where $i = 1, 2, \ldots, NP$ and $j = 1, 2, \ldots, D$. $CR \in (0, 1)$ is the crossover rate that controls the probability of creating components for the trial vector $\mathbf{u}$ from the mutant vector $\mathbf{v}$. The index j_{rand} is an integer randomly chosen from the set $\{1, 2, \ldots, NP\}$, ensuring that at least one component of the trial vector is provided by the mutated vector.

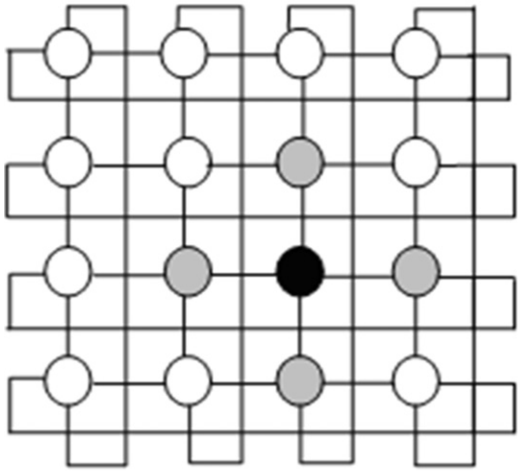

Fig. 1. von Neumann topology.

2.3. Selection

The solution surviving to the next generation is selected from the target individual and its corresponding trial vector according to the following rule:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if} f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases}, \quad (3)$$

where $f(\cdot)$ is the objective function to be minimized. In fact, the selection scheme is a one-to-one tournament replacement. The DE trial vector is not compared against all individuals, but only against its counterpart in the current generation.

3. A novel distributed memetic differential evolution: DMDE

3.1. Migration mechanism in distributed DE

In the new distributed differential evolution, the initial population P is divided into m subpopulations and the size of each subpopulation is $NP_i = NP/m$ ($i = 1, 2, \ldots, m$). These subpopulations are arranged by the von Neumann topology [23]. Only those individuals that are close to each other within the topology are allowed to interact during the evolution. In the von Neumann topology structure, subpopulations are denoted as nodes. Each node has four neighbors (see Fig. 1). The current node is displayed in black, and its neighboring nodes are indicated in grey. The node under examination is isolated from all the other white ones. Each node only exchanges information with its neighbors and has no direct effect on other nodes. However, each node provides and receives information within its neighborhood at the same time, thus making information communicated throughout the topology.

The individuals of the ith subpopulation ($P_i(i = 1, 2, \ldots, m)$) in each node use the DE algorithm as the main frame to evolve and participate in the competition to select the best individuals. Every γ generations, the best individual in each node will replace the worst individual of its neighbor nodes. The migration mechanism among nodes enables the information to be communicated in the whole population, which is beneficial to explore the solution space and find better solutions. Since the replacement will generate the same solutions in subpopulations and lower the diversity of the whole population, the search process may stagnate in certain cases. Ishibuchi and Narukawa [24] proposed a multi-island model using either the L-learning or the B-learning for multiobjective knapsack problems. The 30% islands implement the L-learning in a model with ten islands and the rest perform the B-learning. Inspired by the idea of [24], in the information migration process, we integrate the L-learning and B-learning, and achieve their cooperation by analyzing the traits of the two learning mechanisms as described in Section 3.3.

3.2. Memetic DE

Differential evolution (DE) is a reliable and versatile function optimizer. It generates an offspring until an individual with better performance is generated. However, DE can be subjected to stagnation in cases where no offspring individuals outperform the corresponding parents for a large number of generations. In order to prevent stagnation, and more generally, to reach high performance within the DE framework, the DE can be enhanced by means of a proper hybridization with some local search algorithms.

The local search process is achieved by the Hooke–Jeeves algorithm [25] which is simple and efficient. In the Hooke–Jeeves algorithm, a combination of exploratory move and pattern move is made iteratively to search the optimum solution [26]. In the exploratory move, the current point $\mathbf{x}_1$ is perturbed in positive and negative directions with a scalar predefined step size α along every one of the d dimensional axes at a time, and the best point $\mathbf{x}_2$ is recorded. The current point $\mathbf{x}_1$ is changed to the best point $\mathbf{x}_2$ at the end of each perturbation if $f(\mathbf{x}_2) < f(\mathbf{x}_1)$ (for minimization); otherwise, the step size α is reduced to continue the process. The exploratory move terminates until all dimensions are exhausted, and the best points are used to perform the pattern move.

The pattern move acts as follows: after the kth exploratory move, $\mathbf{x}_{k+1}$ is obtained as the current base point. Repeat the successful moves in a combined pattern move, that is:

$$\mathbf{x} = \mathbf{x}_{k+1} + \beta(\mathbf{x}_{k+1} - \mathbf{x}_k), \quad \beta > 0. \quad (4)$$

If the new point $\mathbf{x}$ has better fitness, take it as the new base point.

In DE, the greedy competition selection scheme can lead to stagnation. In view of this, we combine the Hooke–Jeeves local search with the selection scheme of DE. For each individual, when $f(\mathbf{x}_{crossover}) > f(\mathbf{x})$ ($\mathbf{x}_{crossover}$ denotes the solution after the crossover in DE), we employ a probability value $p = g/g_{\max}$ to judge whether the Hooke–Jeeves algorithm is used to update DE, where g is the current generation and $g_{\max}$ is the allowable maximum generation. The Hooke–Jeeves algorithm is applied if and only if $rand(0, 1) < p$; otherwise, no update occurs. So, in the early stage of the memetic DE algorithm, in order to keep population diversity, the Hooke–Jeeves algorithm is rarely used and DE is the main evolution method. In the later evolution stage, the Hooke–Jeeves algorithm has larger opportunity to be applied for enhancing the local search ability and improve the precision of solutions.

3.3. Cooperation of Lamarckian learning and Baldwinian learning

There are two learning mechanisms to combine evolutionary search and local search heuristics. One of the mechanisms is the L-learning. In this mechanism the solutions generated during the course of evolution are fine-tuned. After individual improvement, the solution itself and its associated fitness value are modified. The modified solution is then inserted back into the population for subsequent evolutionary processing. The other mechanism is the B-learning. It is similar to the L-learning in that the evolutionary search is interleaved with local search and the solutions' fitness values are modified by local search. However, instead of inserting the solution obtained by local improvement into the population, the original solution before the application of local search is retained for subsequent evolutionary processing [27]. The results obtained in [28] show that, even if a solution has an undesirable inborn fitness, it may still have a high chance to find the global optimum,

Table 1
The dynamics of *CV* with the solution and fitness.

Schwefel 2.22: $f_2(x) = \sum_{i=1}^{2}|x_i| + \prod_{i=1}^{2}|x_i|, \quad |x_i| < 10$

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$f(\mathbf{x}_1)$	$f(\mathbf{x}_2)$	$f(\mathbf{x}_3)$	$f(\mathbf{x}_4)$	*CV*
$g = 0$	(9.00, 7.82)	(−5.37, 5.24)	(2.13, −0.87)	(−0.28, −9.62)	87.28	38.81	**4.86**	12.61	1.03
$g = 10$									
L	(−1.03, 0.06)	(−1.09, −0.02)	(−1.02, 0.066)	(−1.07, −0.01)	1.16	**1.13**	1.15	1.11	0.01
B	(9.00, 7.82)	(−5.37, 5.24)	(2.13, −0.87)	(−0.28, −9.62)	9.67	36.14	**4.86**	12.55	0.88
$g = 100$									
L	(−1.06, −0.00)	(−1.06, −0.00)	(−1.06, −0.00)	(−1.06, −0.00)	1.06	1.06	**1.06**	1.06	0
B	(9.00, 7.82)	(−5.37, 5.24)	(2.13, −0.87)	(−0.28, −9.62)	9.67	24.91	2.24	**0.81**	1.17

Griewank: $f_7(x) = \frac{1}{4000}\sum_{i=1}^{2}(x_i)^2 - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad |x_i| < 600$

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$f(\mathbf{x}_1)$	$f(\mathbf{x}_2)$	$f(\mathbf{x}_3)$	$f(\mathbf{x}_4)$	*CV*
$g = 0$	(540.15, 469.55)	(−322.63, 314.51)	(128.21, −52.23)	(−16.82, −577.79)	128.51	51.29	**6.39**	84.96	0.76
$g = 10$									
L	(19.00, 52.25)	(8.56, 29.73)	(15.82, 50.62)	(2.49, −34.94)	0.56	0.76	**0.35**	0.60	0.28
B	(540.15, 469.55)	(−322.63, 314.51)	(128.21, −52.23)	(−16.82, −577.79)	40.38	51.29	**6.39**	28.17	0.61
$g = 100$									
L	(18.68, 17.95)	(18.69, 17.95)	(21.89, 13.55)	(21.89, 13.55)	0.18	0.18	0.18	**0.18**	0.01
B	(540.15, 469.55)	(−322.63, 314.51)	(128.21, −52.23)	(−16.82, −577.79)	40.36	35.30	**0.09**	1.47	1.11

while the Lamarckian strategy, though faster, may lead to a local optimum.

These characteristics of the L-learning and the B-learning prompt us to investigate the efficiency of the cooperation of these two learning mechanisms. To realize this cooperation, we employ the coefficient of variance (*CV*) [29], a statistical measure of the dispersion of data points in a data series. It is calculated as follows:

$$CV = \frac{\sigma}{\mu + r_p} \tag{5}$$

where σ and μ represent the standard deviation and mean of a group data, respectively and r_p is a small non-negative number to ensure that *CV* has a meaningful denominator when $\mu = 0$. The coefficient of variation represents the ratio of the standard deviation to the mean, and it is a useful statistic for comparing the degree of variation from one data series to another, even if the means are drastically different from each other. The higher the *CV*, the greater the dispersion in the variable. The lower the *CV*, the smaller the residuals relative to the predicted value.

In order to illustrate the dynamics of *CV*, Table 1 shows the change of *CV* with the solution and fitness for the 2-dimensional Schwefel 2.22 function and Griewank function.

The former function is unimodal and the later is multimodal [30]. DE/rand/1bin is employed as the evolution frame. The parameters *F* and *CR* are set as 0.5 and 0.9 respectively according to the literature [31]. Besides, we choose $NP = 4$ for ease of comprehension. The best fitness values of the L-learning and B-learning in the given generation are highlighted in boldface.

The L-learning and B-learning (L and B in Table 1) operate on the initial individuals respectively. From the values of the individuals $\mathbf{x}_i (i = 1–4)$, their fitness $f(\mathbf{x}_i)$ and the *CV* after L-learning and B-learning, some conclusions can be achieved as follows:

- The *CV* values after L-learning are smaller than the ones after B-learning, e.g. $0.01 < 0.88$ and $0 < 0.17$ for f_2; $0.28 < 0.61$ and $0.01 < 0.11$ for f_7.
- At the earlier stage of DE, e.g. when $g = 10$, the L-learning can gain better fitness results than B-learning, e.g. $1.13 < 4.86$ for f_2 and $0.35 < 6.39$ for f_7.
- At the later stage of DE, e.g. when $g = 100$, the B-learning can gain better fitness results than L-learning, e.g. $0.81 < 1.06$ for f_2 and $0.09 < 0.18$ for f_7.

The conclusions acquired from Table 1 above are in accordance with the ones in [28]. So, *CV* can be used as a diversity index to reflect the influence of L-learning and B-learning on the individuals.

In our method, the cooperation scheme is applied in the hybridization of DE and the Hooke–Jeeves algorithm as well as the migration among subpopulations. In the process of the hybridization between DE and Hooke–Jeeves, we calculate the *CV* of the fitness before and after the hybridization, respectively. Denote the former by CV_1 and the later by CV_2. If $CV_1 < CV_2$, the population updated by the hybridization of DE and the Hooke–Jeeves algorithm has the greater dispersion. On the occasion, more undesirable individuals are likely to slow down the evolution. So, the L-learning is expected to find efficient solutions faster in favor of exploitation. If $CV_1 \geq CV_2$, the population after the hybridization has smaller or identical dispersion. In this situation, it is potential to find better solutions but stagnate around local optima. So, the B-learning is required to keep the diversity of population to benefit exploration. Regarding the information migration in distributed DE, each subpopulation exchanges information with its neighbors only if the new immigrant has a better performance than the worst one in the target subpopulation, i.e. $f(\mathbf{x}^{k}_{best}) < f(\mathbf{x}^{k+n}_{worst})$, $n = 1–4$. The cooperation of the L-learning and the B-learning is also considered in the migration process which can be regarded as a way of social learning at subpopulation level. Denote the *CV* of the fitness before the migration by CV_3 and that after the migration by CV_4, respectively. The cooperation policy is akin to that employed in the process of the hybridization between DE and Hooke–Jeeves. If $CV_3 < CV_4$, the subpopulation that has exchanged the information with its neighbors obtains greater dispersion. It is difficult to find competitive solutions in short time and the whole population will evolve slowly. Thereby, the L-learning is executed to replace the worst individuals in each subpopulation with the best ones in its neighbors. In this situation, the subpopulation not only retains its best individual but also absorbs the best individuals of its neighbors. It is beneficial to accelerate the evolution and enhance the exploitation. If $CV_3 \geq CV_4$, the subpopulation through the information migration has smaller or identical dispersion and the whole population contains more similar or identical individuals, which is disadvantageous to the exploration of the solution space. Under this circumstance, the B-learning takes effect, that is, only the fitness values of the worst individuals in each subpopulation are replaced in the migration. Due to the fitness sharing through migration, the worst individuals in each subpopulation before migration may not be the worst ones in subsequent generations. In this way, the fast convergence toward the so-far-best solutions is moderated, which helps to maintain population diversity and explore more superior solutions in the succeeding evolution.

For the sake of clarity, the pseudo-code illustrating the principle of DMDE is shown in Fig. 2.

The following two examples are given to elaborate on the cooperation of L-learning and B-learning in the hybridization of DE and Hooke–Jeeves algorithm as well as the migration among subpopulations respectively. The 2-dimensional Schwefel 2.22 function is applied and 4 individuals are acquired randomly for conciseness.

Example 1. Referring to Fig. 3, the hybridization acts on every individual sequentially in accordance with the conditions $f(\mathbf{x}_{crossover}) > f(\mathbf{x})$ and $rand(0,1) < p$ shown in Section 3.2. The three parameters of DE were set as the ones corresponding to Table 1. The step size α was set to be $0.25 \times b$ (b: the variable value of the current dimension for the base point $\mathbf{x}$) and the coefficient β was set to be 1 in Hooke–Jeeves algorithm (H–J in Fig. 3) in accordance with the suggestions given in [33].

For $\mathbf{x}_1$ and $\mathbf{x}_4$, the hybridization conditions are not satisfied and they retain the results of DE. The hybridization conditions are achieved for $\mathbf{x}_2$ and $\mathbf{x}_3$. So the B-learning is adopted for $\mathbf{x}_2$ due to $CV_1 > CV_2$ and the L-learning is considered for $\mathbf{x}_3$ because of $CV_1 < CV_2$. From the results in Example 1, it can be seen that the cooperation of L-learning and B-learning in the hybridization of DE and Hooke–Jeeves algorithm leads to a better fitness (6.03e−06) than DE alone (9.68).

Example 2. As shown in Fig. 4, we assume that 4 neighbors exist for the current kth subpopulation in the migration process among the subpopulations. The small non-negative number r_p was set to 2e−16 for higher computation accuracy. The situation of the neighbors receiving the information is taken into consideration. The individual $\mathbf{x}_2$ has the best fitness (11.14) for the current subpopulation. The migration condition is that the best fitness of the subpopulation is smaller than the worst one of its neighbor, that is, $f(\mathbf{x}^k_{best}) < f(\mathbf{x}^{k+n}_{worst})$, $n = 1$–4.

For neighbor 1, individual $\mathbf{x}_3$ is the worst one. Due to its fitness 55.93 > 11.14, the migration acts on it. So, the B-learning takes effect because of $CV_3 < CV_4$, that is, only its fitness is changed from 55.93 to 11.14 but the solution $\mathbf{x}_3$ is retained. For neighbor 2, the worst individual is $\mathbf{x}_1$. Similarly, this neighbor undergoes the migration. The worst solution and its fitness are replaced by the best ones in the current subpopulation. The migration and cooperation are not achieved for neighbor 3 since their conditions are not satisfied. Neighbor 4 has the same situation as neighbor 1 and only the worst fitness is changed from 58.27 to 11.14.

After the migration and cooperation, it can be seen that the worst individuals in the neighbors are improved, which provides a better foundation for the subsequent evolution.

4. Numerical experiment and analysis

To demonstrate the efficiency of DMDE, we performed the following numerical experiments. The test problems involved are presented below [30,32,34,35].

$$\text{Sphere}: \quad f_1(x) = \sum_{i=1}^{D} x_i^2, \quad |x_i| < 1$$

$$\text{Schwefel 2.22}: \quad f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|, \quad |x_i| < 10$$

$$\text{Quadric}: \quad f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2, \quad |x_i| < 1$$

$$\text{Noisy quartic}: \quad f_4(x) = \sum_{i=1}^{D} i \cdot x_i^4 + U(0,1), \quad |x_i| < 1.28$$

$$\text{Ackley}: \quad f_5(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \cdot \sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi \cdot x_i) \right) + 20 + e, \quad |x_i| < 1$$

$$\text{Alpine}: \quad f_6(x) = \sum_{i=1}^{D} |x_i \sin x_i + 0.1 x_i|, \quad |x_i| < 10$$

$$\text{Griewank}: \quad f_7(x) = \frac{1}{4000} \sum_{i=1}^{D} (x_i)^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1, \quad |x_i| < 600$$

$$\text{Rastrigin}: \quad f_8(x) = \sum_{i=1}^{D} (x_i^2 - 10 \cos(2\pi x_i) + 10), \quad |x_i| < 5.12$$

$$\text{Rosenbrock}: \quad f_9(x) = \sum_{i=1}^{D} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, \quad |x_i| < 2.048$$

$$\text{Schwefel}: \quad f_{10}(x) = 418.9829 \times D - \sum_{i=1}^{D} \sin(|x_i|^{1/2}), \quad |x_i| < 500$$

Functions f_1–f_3 are unimodal and function f_4 is a noisy quadratic function, where $U(0,1)$ is a uniformly distributed random variable in [0,1]. Function f_9 is a multimodal function when $D > 3$. Functions f_5–f_8 and f_{10} are multimodal functions, where the number of local minima increases exponentially with the problem dimension. The minimum value of all the test functions is zero. The dimension of each test function is set as $D = 50$.

4.1. Parameters setting of DMDE: m and γ

DMDE is implemented on the basis of DE/rand/1/bin. It will be terminated until the number of function evaluations (NFE) reaches 1,000,000. The parameters of classic DE were set as $NP = 400$, $F = 0.5$

```
initialize population
    for individual i = 1 : NP
        compute the objective value f(x_i)
    end
while the stopping condition is not met do
    for generation g = 1 : g_max
        for subpopulation k = 1 : m
            for individual i = 1 : NP/m
                differential mutation -> v_{i,g}
                crossover -> u_{i,g}
                if f(x_crossover) > f(x)
                    if rand(0,1) < g/g_max
                        Hooke-Jeeves -> x_{i1,g}
                        compute CV_1 and CV_2
                        [Cooperation of L-learning and B-Learning in the hybridization of DE and the Hooke–Jeeves algorithm]
                        if CV_1 < CV_2
                            x_{i,g} = x_{i1,g} ; f(x_{i,g}) = f(x_{i1,g})  L learning
                        else
                            x_{i,g} = x_{i,g} ; f(x_{i,g}) = f(x_{i1,g})  B learning
                        end if
                    end if
                end if
            end for
        end for
        if g is an integer multiple of γ (learning period)
            for k = 1 : m
                for n=1:4
                    if f(x^k_best) < f(x^{k+n}_worst)
                        select and copy x^k_best
                        compute CV_3 and CV_4
                        [Cooperation of L-learning and B-Learning in the migration among subpopulations.]
                        if CV_3 < CV_4
                            x^{k+n}_worst = x^k_best ; f(x^{k+n}_worst) = f(x^k_best)  L learning
                        else
                            x^{k+n}_worst = x^{k+n}_worst ; f(x^{k+n}_worst) = f(x^k_best)  B learning
                        end if
                    end if
                end for
            end for
        end if
    end for
end while
```

Fig. 2. The pseudo-code of DMDE.

and $CR = 0.9$ according to the suggestions given in [35]. The parameters of the Hooke–Jeeves α and β were set the same as in Example 1. The small non-negative number r_p was set the same as in Example 2. Regarding each test function, 30 independent runs were performed.

DMDE has two crucial parameters m and γ: m is the number of subpopulation and γ is the migration period, that is, the interval between two migrations. The parameter m influences the size of the subpopulation. The smaller m, the larger the subpopulation size. γ dominates how often the migration occurs. The smaller γ, the more frequent the migration. To determine the value of the two parameters, we consider three candidate values for both: $m = 5$, 20, 50, and $\gamma = 10$, 100, 500. 9 combinations can be formed for the two parameters. Statistical results about the discovered minimum of the objective values in 30 independent runs regarding the test functions f_5 and f_8 are shown in Fig. 5.

It can be seen that the combination $m = 5$, $\gamma = 100$ obtains significantly better results in contrast to the other 8 combinations. This parameter setting not only produces the best average results w.r.t. the discovered minimal objective values but also contributes to a better distribution of the solutions.

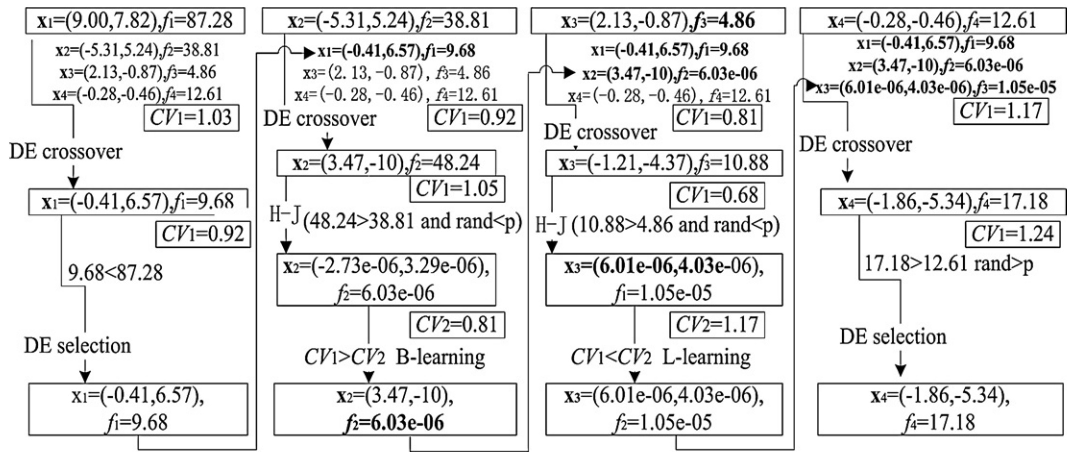

Fig. 3. The cooperation of L-learning and B-learning in the hybridization of DE and Hooke–Jeeves.

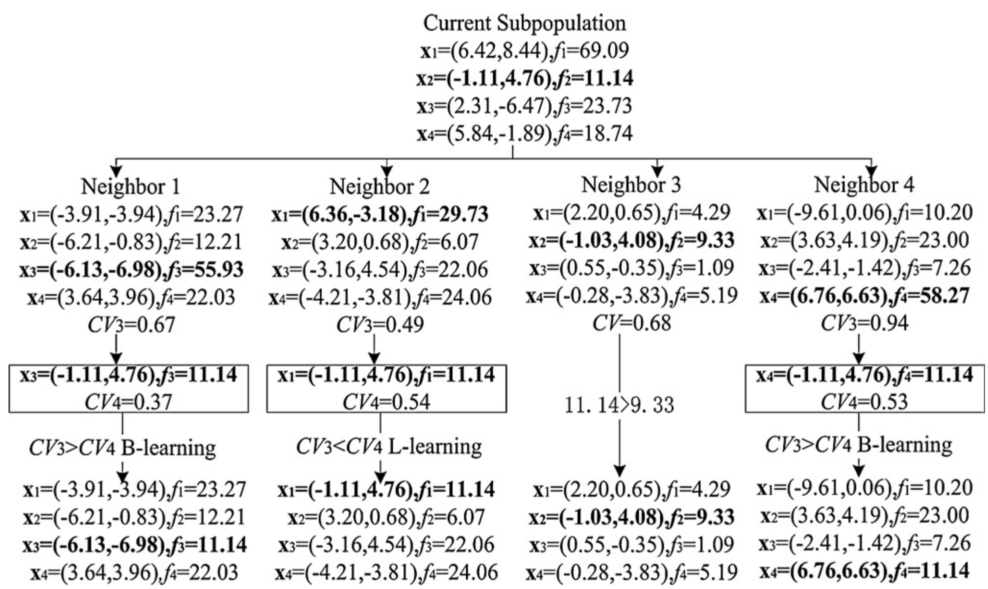

Fig. 4. The cooperation of L-learning and B-learning in the migration among subpopulations.

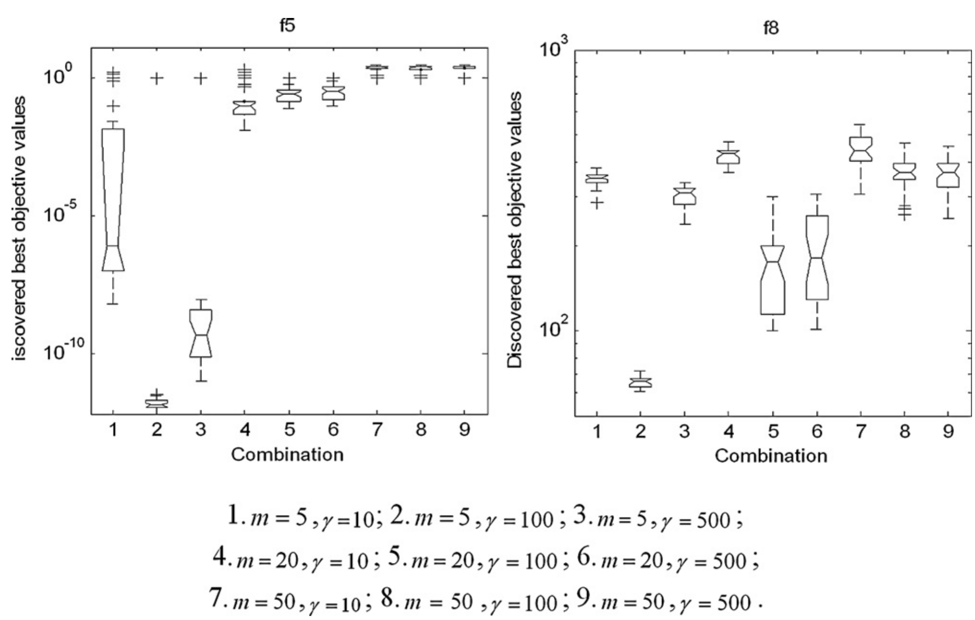

Fig. 5. The performance of DMDE with 9 combinations of the parameters m and γ.

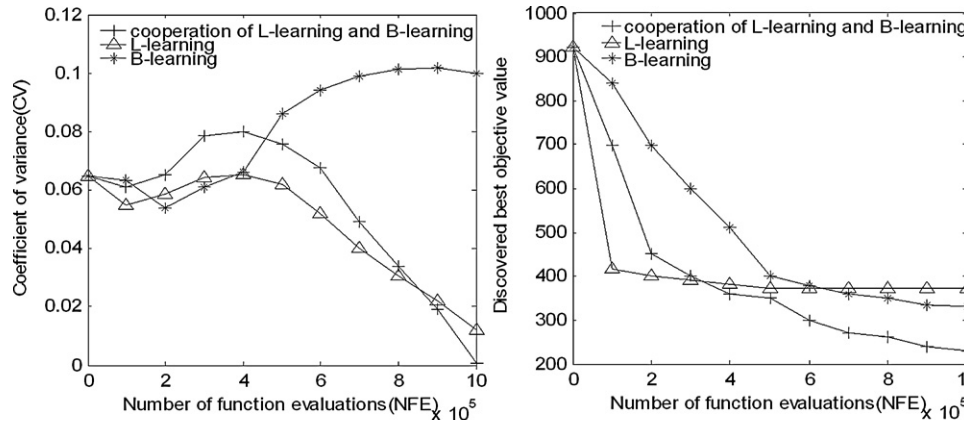

Fig. 6. The trend of *CV* in the cooperation of L-learning and B-learning.

4.2. Analysis of DMDE

For a better understanding of the cooperation of two learning mechanisms, Fig. 6 shows the trend of the *CV* and the best objective value for f_8 when we applied L-learning, B-learning and the cooperation of L-learning and B-learning, respectively.

From Fig. 6, we can see that the population diversity decreases rapidly and the population easily leads to stagnation when we use the L-learning alone. On the contrary, although the population diversity with the B learning increases with the evolution and better solutions can be obtained, the progress of evolution is slow. Nevertheless, when we combine the B-learning with the L-learning, the population is improved gradually as it is diversified at the early evolution stage and the solutions are renovated continually. So, it is reasonable to coordinate the L-learning and B-learning through regulating the value of *CV*. Meanwhile, the cooperation strategy achieves better results than L-learning and B-learning alone.

Fig. 7 explains the advantage of integrating the hybridization strategy and the multi-population strategy into DE. The convergence rate and the average running time (t) for 30 times of these schemes are shown in Fig. 7. The average fitness values of all schemes were recorded by 11 points of 30 times for f_5 and f_8.

As shown in Fig. 7, the performance of the proposed DMDE outperforms the basic DE, the distributed DE, and the hybrid based on DE and Hooke–Jeeves in terms of the minimal fitness value, the convergence rate and the average time. Nevertheless, the later two schemes lead to better results than the basic DE. So, it is beneficial for the proposed DMDE to integrate the distributed DE scheme and the cooperation of two learning mechanisms.

Fig. 8 shows the trend of the *CV*, the best and the worst fitness values of the whole population [Fig. 8(a)] and one subpopulation randomly chosen from 5 subpopulations [Fig. 8(b)] for f_8. These values were recorded in a quarter of migration period after every migration.

From Fig. 8, it can be found that the values of *CV* increase rapidly at the beginning of the evolution and decrease continually in the later stages of the operation. It indicates that the DMDE scheme is more explorative at the beginning of the evolution and subsequently turns into exploitation. At the same time, the best and the worst fitness values are improved stably along with the increase of the number of fitness evaluation in population and subpopulations.

4.3. Comparison with other distributed DE algorithms

4.3.1. Computational complexity comparison

For IBDDE, DDE and FACPDE, their main operations include the operations of the basic DE and the migration mechanism. And for DMDE, its main operations include the hybrid DE and the migration mechanism.

For one function evaluation, the time cost of the basic DE is caused by the time for the differential mutation t_m, the time for the crossover t_c and the time for the selection t_s. Each individual (solution) is D-dimensional. As a result, the time complexity is $O(D)$ for the differential mutation and crossover. Nevertheless, since the improvement of an individual is not always performed, the worst-case time complexity of the selection is $O(D)$. For

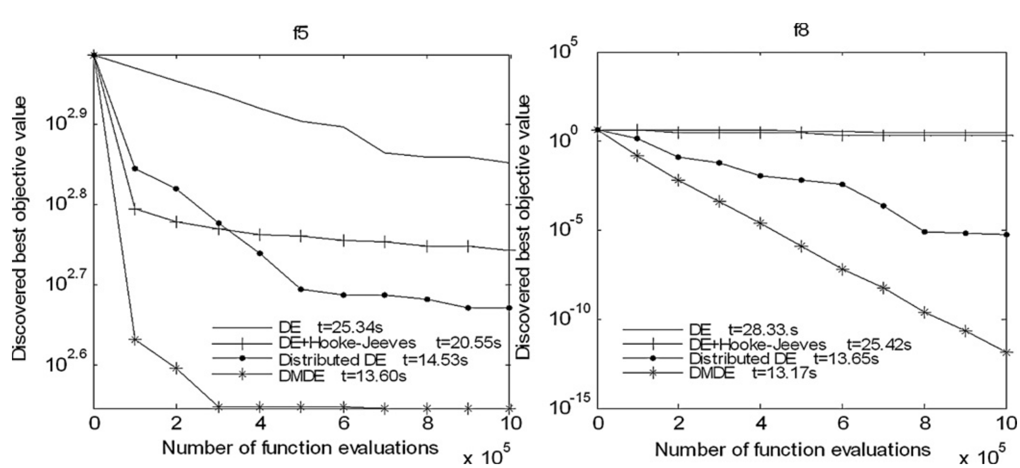

Fig. 7. The performance of DMDE compared with three schemes.

Table 2
The time complexity of the IBDDE, DDE, FACPDE and DMDE.

Main operations	Time	Time complexity	
		IBDDE DDE FACPDE	DMDE
DE			
Differential mutation	t_m	$O(D)$	$O(D)$
Crossover	t_c	$O(D)$	$O(D)$
Selection	t_s	$O(D)$	$O(D)$
Hybridization			
Hooke–Jeeves	t_e, t_p	–	$O(2D+1)$
CV_1CV_2	$t_{CV_{12}}$	–	$O(t_{CV_{12}})$
Function evaluation	t_f	$O(t_f)$	$O(t_f)$
Migration mechanism			
Migration	t_M	$O((g_{\max}/\gamma)^* m^2 * t_M)$	$O((g_{\max}/\gamma)^* m^2 * t_M)$
CV_3CV_4	$t_{CV_{34}}$	–	$O(t_{CV_{34}})$
Total		$O(NFE^* t_f + (NFE-NP)^* D + (g_{\max}/r)^* m^2 * t_M)$ $g_{\max} = (NFE-NP)/NP$	$O(NFE * t_f + ((NFE-NP)/(1+\eta)) * D + (((NFE-NP) * \eta/(1+\eta)) * ((2D+1) + t_{CV_{12}})) + ((g_{\max}/r) * m^2 * (t_M + t_{CV_{34}})))$ $g_{\max} = ((NFE-NP)/(1+\eta) * NP)$

IBDDE, DDE and FACPDE, individuals excepting initial NP ones involve the operations of DE. NFE denotes the maximal number of function evaluations. The number of the individuals manipulated by DE is $NFE-NP$. Since there are NP individuals in every generation, the number of these individuals can also be expressed as $g_{\max} * NP$, where $g_{\max} = (NFE-NP)/NP$. So, the time complexity of the basic DE is $O((NFE-NP)^* D)$ for IBDDE, DDE and FACPDE. In addition, the number of the migration is $g_{\max}/\gamma$ in the whole evolution. For each subpopulation, the migrated individual and the replaced individuals of its neighbor subpopulations need to be chosen in the process of migration. Since there are m subpopulations in the whole population, the time complexity of the migration is $O((g_{\max}/r)^* m^2 * t_M)$ for IBDDE, DDE and FACPDE, where t_M denotes the time for one migration. The whole time cost of the algorithm can be represented by the sum of the time for total function evaluations and the time for the operation corresponding to the function evaluation. In a word, the time complexity of the IBDDE, DDE and FACPDE is $O(NFE^* t_f + (NFE-NP)^* D + (g_{\max}/r)^* m^2 * t_M)$, where t_f is the time for one function evaluation.

For DMDE, the time complexity of the differential mutation, crossover and selection is also $O(D)$ for one function

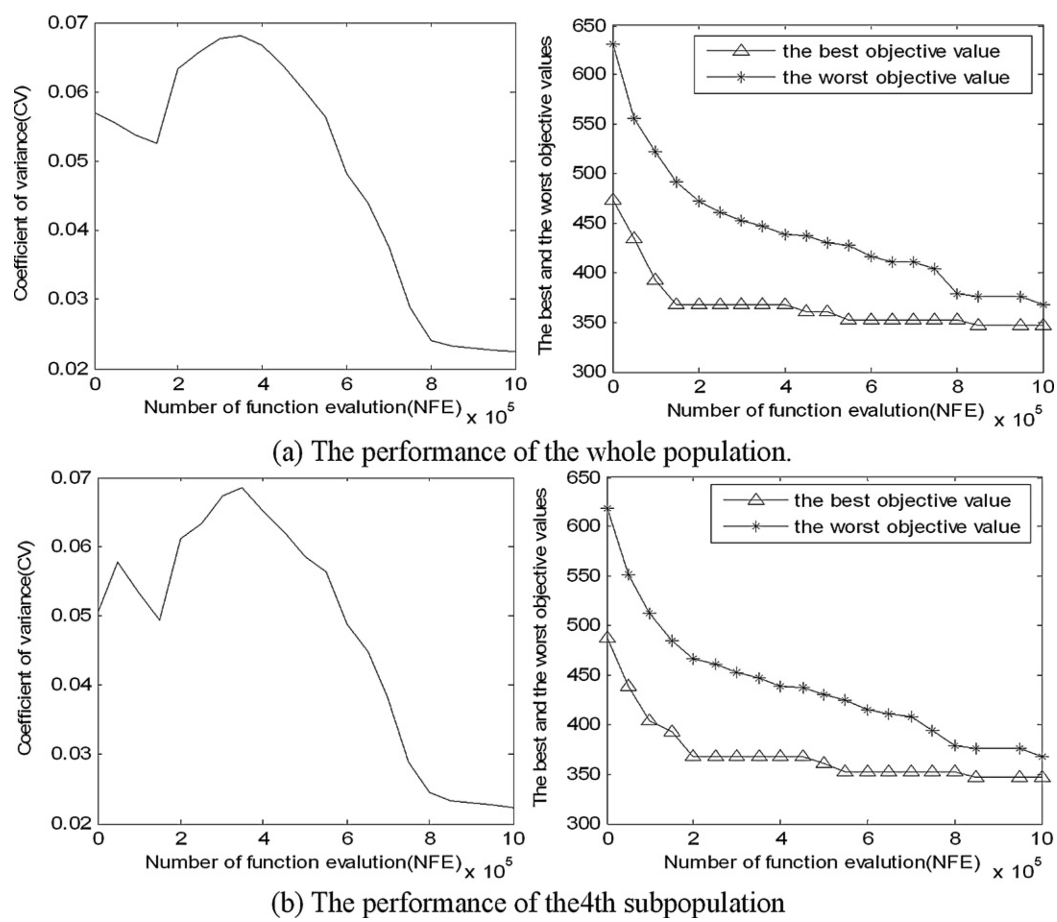

Fig. 8. The trend of the CV, the best and the worst fitness value of the whole population and the subpopulation for f_8. (a) The performance of the whole population. (b) The performance of the 4th subpopulation.

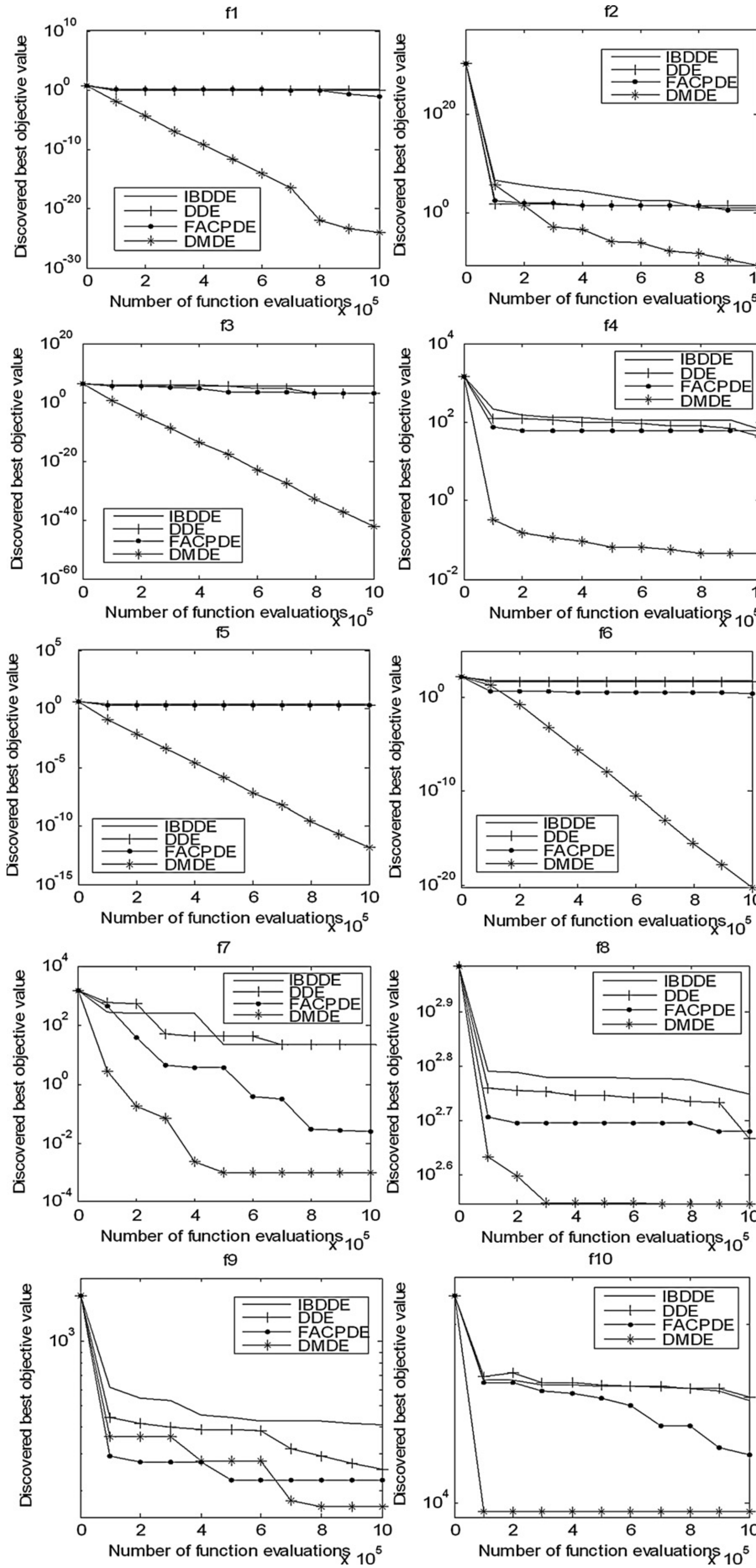

Fig. 9. Convergence plots.

evaluation, which is the same as those in the compared algorithms. However, in DMDE, the basic DE can be integrated with the Hooke–Jeeves algorithm when the conditions are satisfied. Since the basic DE and the Hooke–Jeeves execute the function evaluation, the equation $NP * g_{\max} + \eta * NP * g_{\max} = NFE - NP$, where $0 \leq \eta \leq 1$ is established, then $g_{\max} = ((NFE - NP)/(1+\eta) * NP)$ is obtained in DMDE where $NP * g_{\max}$ is the number of individuals manipulated by DE; $\eta * NP * g_{\max}$ is the number of individuals manipulated by Hooke–Jeeves. As a result, the time complexity of the basic DE is $O((NP * g_{\max}) * D) = O((NP * (NFE - NP)/((1+\eta) * NP)) * D) = O(((NFE - NP)/(1+\eta)) * D)$ for DMDE. For one function evaluation, the time cost of Hooke–Jeeves is caused by the time for the exploratory move t_e and the time for the pattern move t_p. Since the current point is perturbed in positive and negative directions in the exploratory move and the new point is obtained by the pattern move [see (4)], the time complexity is $O(2D+1)$ for the Hooke–Jeeves. So, $O((\eta * NP * g_{\max}) * (2D+1)) = O((\eta * NP * (NFE - NP)/((1+\eta) * NP)) * (2D+1)) = O(((NFE - NP) * \eta/(1+\eta)) * (2D+1))$ is the time complexity of Hooke–Jeeves in DMDE. In addition, the time $t_{CV_{12}}$ for CV_1 and CV_2 is calculated in the hybridization. In the migration, except t_M, the time cost includes $t_{CV_{34}}$ for CV_3 and CV_4. Therefore, the time complexity of the migration is $O((g_{\max}/r) * m^2 * (t_m + t_{CV_{34}}))$ for DMDE. In a word, the time complexity of DMDE for the whole evolution is $O(NFE * t_f + ((NFE - NP)/(1+\eta)) * D + (((NFE - NP) * \eta/(1+\eta)) * ((2D+1) + t_{CV_{12}})) + ((g_{\max}/r) * m^2 * (t_M + t_{CV_{34}})))$. Table 2 shows the time complexity of IBDDE, DDE, FACPDE and DMDE.

It is notable that, compared with the time for function evaluation, the time for other operations take up a small proportion in one running iteration. Table 3 shows the comparison of the time for function evaluation and other operations in one running iteration, where the time for all operations in one running iteration is corresponding to those in Table 2 respectively (e.g. $T_m \rightarrow t_m$). Therefore, these algorithms almost have the same time cost in optimizing the same test function. Meanwhile, the space complexity of DMDE is $O(m \times (NP/m) \times D) = O(NP \times D)$.

Table 3
The comparison of the time for function evaluation and other operations (unit: s) in one running iteration.

Algorithm	Fun. f_8 t								
	T_m	T_c	T_s	T_M	T_e	T_p	$T_{CV_{12}}$	$T_{CV_{34}}$	T_f
DDE	1.13	1.41	0.72	0.57	1.21	1.33	0.41	0.25	12.58
IBDDE	1.75	1.42	0.71	0.35	1.17	1.75	0.42	0.27	12.31
FACPDE	1.45	1.39	0.70	0.69	1.15	1.45	0.39	0.23	11.43
DMDE	1.31	1.46	0.74	0.31	1.13	1.31	0.46	0.24	11.39

From the above analysis, DMDE has no obvious difference with the other three algorithms in terms of computational complexity.

4.3.2. Results and performance comparison

To prove the viability of the DMDE and test its performance, the DMDE was compared with IBDDE [13], DDE [14] and FACPDE [16]. For a fair comparison, the termination criteria, the applied basic DE variant and its three control parameters for the three algorithms were all set as same as the DMDE mentioned above. 30 independent runs were performed for each algorithm. The other parameters of the three algorithms were set as follows.

IBDDE was run with a population of 400 individuals, which is divided into 5 subpopulations of 80 individuals each. The other parameters were chosen according to the values in Apolloni et al. [13], and the subpopulations exchange one individual every 100 generations. The individual to be migrated is randomly chosen from the current subpopulation. Incoming individuals from other subpopulations replace a randomly chosen local individual, only if the former is better. A unidirectional ring topology was employed.

In DDE, the population holds 400 individuals, which is divided into 16 subpopulations of 25 individuals. Following the suggestions in Falco et al. [14], a torus mesh topology was

Table 4
Comparison in terms of minimum, mean ± standard deviation and the average running time (s).

Fun.	IBDDE	DDE	FACPDE	DMDE
f_1	2.74e+00	1.34e+01	2.04e+00	**4.93e−25**
	3.75e+00 ± 8.53e−02	2.41e+01 ± 1.04e−01	2.69e+00 ± 7.45e−02	**1.00e−23 ± 2.00e−24**
	12.35s	12.34s	12.13s	**12.03s**
f_2	1.23e+02	7.57e+01	8.82e+01	**8.04e−11**
	1.39e+02 ± 1.44e+00	5.36e+01 ± 1.20e+00	9.23e+03 ± 3.49e+03	**4.58e−10 ± 9.10e−11**
	12.78s	12.75s	**12.62s**	12.64s
f_3	4.81e+04	5.73e+03	9.67e+03	**1.45e−35**
	7.59e+04 ± 2.86e+03	2.06e+04 ± 2.11e+03	3.48e+04 ± 1.95e+03	**2.38e−35 ± 1.54e−33**
	14.25s	14.27s	14.26s	14.26s
f_4	5.02e+01	1.96e+01	3.79e+01	**1.77e−02**
	9.49e+01 ± 4.66e+00	4.24e+01 ± 2.92e+00	5.59e+01 ± 2.28e+00	**2.95e−02 ± 1.75e−03**
	13.89s	13.61s	**13.55s**	13.60s
f_5	2.26e+00	1.82e+00	1.82e+00	**6.55e−13**
	2.64e+00 ± 0.02e+00	2.15e+00 ± 0.04e+00	2.31e+00 ± 0.03e+00	**1.56e−12 ± 1.34e−13**
	13.47s	13.38s	13.36s	**13.30s**
f_6	6.05e+01	4.30e+01	4.56e+01	**2.97e−21**
	6.58e+01 ± 5.7e−01	5.36e+01 ± 1.20e−00	5.26e+01 ± 6.34e−01	**3.86e−15 ± 8.11e−15**
	13.38s	13.61s	**13.35s**	13.40s
f_7	9.38e−02	4.30e−02	6.14e−02	**0**
	1.17e−01 ± 2.23e−03	7.31e−02 ± 3.36e−03	9.07e−02 ± 2.67e−03	**8.21e−04 ± 4.71e−04**
	13.74s	13.79s	**13.60s**	13.65s
f_8	5.09e+02	3.74e+02	4.24e+02	**2.74e+02**
	5.54e+02 ± 4.41e+00	4.50e+02 ± 4.99e+00	5.04e+02 ± 4.97e+00	**3.41e+02 ± 3.93e+00**
	13.31s	13.22s	**13.15s**	13.17s
f_9	2.10e+03	6.53e+02	1.32e+03	**2.84e+02**
	2.89e+03 ± 8.6e+01	1.26e+03 ± 5.97e+01	1.90e+03 ± 5.19e+01	**3.47e+02 ± 3.51e+00**
	12.52s	12.42s	12.30s	**12.28s**
f_{10}	1.41e+04	1.32e+04	1.41e+04	**8.70e+03**
	1.51e+04 ± 6.85e+01	1.44e+04 ± 7.63e+01	1.50e+04 ± 6.70e+01	**1.04e+04** ± 1.35e+02
	14.02s	13.96s	13.60s	**13.55s**

employed. Each subpopulation sends a copy of its best individual to replace the worst ones of its neighbors every 5 generations.

FACPDE was run with a population of 400 individuals, which is divided into 5 subpopulations of 80 individuals. According to the suggestions in [16], the migration occurs if *rand* < 0.2 every generation. The individual with the best fitness is duplicated and then replaces a randomly selected individual of the neighbor in a unidirectional ring.

Similar to IBDDE and FACPDE, DMDE was run with a population of 400 individuals, which is divided into 5 subpopulations of 80 individuals each. The migration period was set to $\gamma = 100$. The best individual in each subpopulation is duplicated and replaces the worst one of its neighbors in the von Neumann topology (see Figs. 1 and 4).

Table 4 shows the results obtained by IBDDE, DDE, FACPDE and the proposed DMDE to solve 50-dimensional problems f_1–f_{10}. Regarding each test problem, each algorithm ran independently 30 times. The minimum, the mean plus standard deviation of the objective values found and the average running time for 30 runs are listed in Table 4. The best results are highlighted in boldface.

Results in Table 4 show that DMDE has a very good performance for the test problems f_1–f_7, since it can find those solutions with the best minimum, average and standard deviation values and outperform the other three algorithms with a large margin. DMDE provides the best minimal and average values among the algorithms for f_{10}. As for f_8 and f_9, DMDE achieves the best minimum, average and standard deviation. For one running iteration, on average, DMDE takes up the least amount of time for f_1, f_5, f_9 and f_{10} and approximate time for other functions compared with the other three algorithms. In this sense, DMDE is a very efficient algorithm for various test problems.

Fig. 9 shows the average convergence trends of the compared algorithms. All algorithms were run 30 times, and 11 points were recorded every time. For every point, the results were obtained by averaging the fitness value of 30 times.

Fig. 9 shows the results of the average performance trends. The DMDE algorithm is obviously superior to IBDDE, DDE and FACPDE. The DMDE has good convergence speed during the early stages of the evolution for f_4 and f_{10}, and it detects high quality solutions during the initial generations. The performance gap between DMDE and the second best algorithm is much larger for f_1, f_3, f_5 and f_6. Besides, regarding f_3, f_5 and f_6, DMDE has a great ability to evolve the discovered objective value even if the number of function evaluations reaches 10^6. Although DMDE converges slower than DDE and FACPDE for f_2 and slower than FACPDE for f_9, it continuously improves on its solutions and outperforms the other algorithms at the end of evolution.

The comparison shows that the proposed DMDE is capable of tackling both unimodal and multimodal problems. The DMDE is very efficient and has obvious advantages over the three state-of-the-art distributed DE algorithms.

5. Conclusion

A novel distributed memetic differential evolution incorporating two learning mechanisms, namely DMDE, was presented in this paper. The proposed method is inspired by two strategies: the population structure often used in distributed DEs and the hybridization strategy often adapted in MAs. The former strategy divides the initial population into multiple subpopulations according to the von Neumann topology and realizes the periodical information exchange by migration. And the latter idea takes DE as an evolutionary frame that is assisted by Hooke–Jeeves algorithm to balance exploration and exploitation. In the evolution process, the characteristics of the Lamarckian learning and Baldwinian learning are analyzed and the two learning mechanisms are coordinated according to the coefficient of variance. The cooperation strategy was applied in the process of information migration among subpopulations as well as the hybridization between Hooke–Jeeves and DE. Experimental results demonstrate that the cooperation strategy is effective and can achieve good performance. The DMDE was compared with three distributed DE algorithms recently proposed in literature. Numerical results show that DMDE has an excellent performance in terms of solution quality and convergence speed for all problems considered than other distributed DE algorithms.

Acknowledgements

This work is supported by the National Science Fund for Distinguished Young Scholars (Grant No. 60925011), the National Research Fund entitled Optimization and Fault Diagnosis in Complex Dynamic Environment (Grant No. 9140A17051010BQ0104) and the Youth Fund of Taiyuan University of Science and Technology (Grant Nos. 20113003, 20113005).

References

[1] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[2] F.-C. Chang, H.-C. Huang, A refactoring method for cache-efficient swarm intelligence algorithms, Information Sciences (2010), doi:10.1016/j.ins.2010.02.025.
[3] C.H. Hsu, W.J. Shyr, K.H. Kuo, Optimizing multiple interference cancellations of linear phase array based on particle swarm optimization, Information Hiding and Multimedia Signal Processing 1 (4) (2010) 292–300.
[4] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 27–54.
[5] Z.H. Cal, W.Y. Gong, C.X. Ling, H. Zhang, A clustering-based differential evolution for global optimization, Applied Soft Computing 11 (1) (2011) 1363–1397.
[6] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 526–553.
[7] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[8] B. Xin, J. Chen, Z.H. Peng, F. Pan, An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization, Science China Information Sciences 53 (5) (2010) 980–989.
[9] B. Xin, J. Chen, J. Zhang, H. Fang, Z.H. Peng, Hybridizing differential evolution and particle swarm to design powerful optimizers: a review and taxonomy, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews (2011), doi:10.1109/TSMCC.2011.2160941, Available online: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5977033.
[10] J. Olensek, T. Tuma, J. Puhan, Á. Bűrmen, A new asynchronous parallel global optimization method based on simulated annealing and differential evolution, Applied Soft Computing 11 (1) (2011) 1481–1489.
[11] D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Parallel differential evolution, in: IEEE Congress on Evolutionary Computation, Portland Mamott Downtown, Portland, USA, 19–23 June, 2004, pp. 2023–2029.
[12] L. Singh, S. Kumar, Parallel evolutionary asymmetric subsethood product fuzzy-neural inference system: an island model approach, in: Proceedings of the International Conference on Computing: Theory and Applications (ICCTA'07), Kolkata, India, 5–7 March, 2007, pp. 282–286.
[13] J. Apolloni, G. Leguizamón, J. García-Nieto, E. Alba, Island based distributed differential evolution: an experimental study on hybrid testbeds, in: Proceedings of the IEEE International Conference on Hybrid Intelligent Systems, Barcelona, Spain, 10–12 September, 2008, pp. 696–701.
[14] I.D. Falco, D. Maisto, U. Scafuri, E. Tarantino, A.D. Cioppa, Distributed differential evolution for the registration of remotely sensed images, in: Proceedings of the IEEE Euromicro International Conference on Parallel, Distributed and Network-based Processing, Naples, Italy, 07–09 February, 2007, pp. 358–362.
[15] D. Izzo, M. Rucinski, C. Ampatzis, Parallel global optimization meta-heuristics using an asynchronous island-model, in: IEEE Congress on Evolutionary Computation, Trondheim, Norway, 18–21 May, 2009, pp. 2301–2308.
[16] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Computing 14 (11) (2010) 1187–1207.
[17] J. Chen, B. Xin, Z.H. Peng, L.H. Dou, J. Zhang, Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization, IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans 39 (3) (2009) 680–691.
[18] N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, IEEE Transactions on Evolutionary Computation 9 (5) (2005) 474–488.

[19] P. Moscato, M. Norman, A memetic approach for the traveling salesman problem implementation of a computational optimization on message passing systems, in: Proceedings of the International Conference on Parallel Computing and Transputer Applications, IOS Press, Amsterdam, 1992, pp. 187–194.
[20] Q.H. Nguyen, Y.S. Ong, M.H. Lim, A probabilistic memetic framework, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 604–623.
[21] K.V. Price, R.M. Storn, J.A. Lampinen, Differential evolution—a practical approach to global optimization, in: Natural Computing Series, Springer, Berlin, 2005.
[22] K.V. Price, R.M. Storn, Differential evolution: a simple evolution strategy for fast optimization, Dr. Dobb's Journal 22 (April (4)) (1997) 18–24.
[23] B. Dorronsoro, P. Bouvry, Improving classical and decentralized differential evolution with new mutation operator and population topologies, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 67–98.
[24] H. Ishibuchi, K. Narukawa, Spatial implementation of evolutionary multiobjective algorithms with partial Lamarckian repair for multiobjective knapsack problems, in: Proceedings of the 5th International Conference on Hybrid Intelligent Systems (HIS'05), Rio de Janeiro, Brazil, 6–9 November, 2005, 2005, pp. 265–270.
[25] R. Hooke, T.A. Jeeves, Direct search solution of numerical and statistical problems, Journal of ACM 8 (2) (1961) 212–229.
[26] S.K. Bath, J.S. Dhillon, D.P. Kothari, Stochastic multiobjective generation allocation using pattern-search method, IEE Proceedings Generation, Transmission and Distribution 153 (4) (2006) 476–484.
[27] K.W.C. Ku, Enhance the Baldwin effect by strengthening the correlation between genetic operators and learning methods, in: IEEE Congress on Evolutionary Computation, Sheraton Vancouver Wall Centre Hotel, Vancouver, Canada, 16–21 July, 2006, pp. 3302–3308.
[28] P.A. Castillo, M.G. Arenas, J.G. Castellano, J.J. Merelo1, A. Prieto, V. Rivas, G. Romero, Lamarckian Evolution and the Baldwin Effect in Evolutionary Neural Networks, 2006, Available online: http://arxiv.org/PS_cache/cs/pdf/0603/0603004v1.pdf arXiv:cs/0603004.
[29] A.N. Akansu, R.A. Haddad, Factorization of the coefficient variance matrix in orthogonal transforms, IEEE Transactions on Signal Processing 39 (3) (1991) 714–718.
[30] Z.H. Cai, W.Y. Gong, C.X. Ling, DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization, Soft Computing 15 (4) (2011) 645–665.
[31] J. Brest, M.S. Maučec, Population size reduction for the differential evolution algorithm, Applied Intelligence 29 (3) (2008) 228–247.
[32] Y.W. Shang, Y.H. Qiu, A note on the extended Rosenbrock function, Evolutionary Computation 14 (1) (2006) 119–126.
[33] I. Moser1, R. Chiong, A Hooke–Jeeves based memetic algorithm for solving dynamic optimisation problems, Computer Science 5572 (2009) 301–309.
[34] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artificial Intelligence Review 33 (1–2) (2010) 61–106.
[35] M. Weber, F. Neri, V. Tirronen, A study on scale factor in distributed differential evolution, Information Sciences 181 (12) (2011) 2488–2511.