Information Sciences 472 (2019) 180–202

Contents lists available at ScienceDirect

Information Sciences

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

Adaptive differential evolution with a Lagrange interpolation argument algorithm

Qiujun Huang[a], Kai Zhang[b,*], Jinchun Song[a,*], Yimin Zhang[b], Jia Shi[a]

[a] School of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China
[b] Equipment Reliability Institute, Shenyang University of Chemical Technology, Shenyang 110142, China

ARTICLE INFO

Article history:
Received 13 February 2018
Revised 28 August 2018
Accepted 1 September 2018
Available online 8 September 2018

Keywords:
Differential evolution
Evolutionary algorithms
Lagrange interpolation
Local search
Mechanism synthesis

ABSTRACT

Differential evolution (DE) is a simple yet powerful evolutionary algorithm that has been used to solve various complex optimization problems in numerous engineering fields. However, DE has some problems, such as premature convergence and sensitivity to parameter settings. To improve the performance of DE and extend its application, an adaptive differential evolution with the Lagrange interpolation argument algorithm (ADELI) is proposed in this paper. To accelerate the convergence speed of DE, a local search with Lagrange interpolation (LSLI) is introduced into DE. LSLI performs a local search in the neighborhood of the best individual in the current generation to enhance the exploitation capability of DE. Meanwhile, an adaptive argument strategy is presented to adaptively determine whether to use LSLI in terms of its performance in the previous generation, which can balance the global exploration capability and the local exploitation capability of ADELI. To verify the feasibility and effectiveness of ADELI, 30 test functions in the CEC 2014 benchmark sets with different dimensions were simulated. Moreover, a path synthesis problem was also optimized. The results demonstrated that ADELI considerably outperforms other EAs in most functions and obtains the most accurate solution among the compared algorithms in the application of path generation.

1. Introduction

Recently, evolutionary algorithms (EAs) have been widely applied to solve optimization problems in various fields such as engineering and science. In contrast to conventional methods based on mathematical programming or formal logics, EAs are found to be more powerful and flexible [43]. However, current EAs may have problems such as the computation becoming trapped in local optima and a slow convergence speed in solving complex optimization problems, such as nonlinear and nondifferentiable problems [47]. There is a need to develop more effective and efficient EAs. To discourage premature convergence and improve the performance of EAs, increasingly more researchers have been devoted to modifying existing EAs or to developing new EAs based on swarm intelligence and genetic evolution. Numerous EAs have been proposed and successfully applied in science and engineering fields. The commonly used EAs include the genetic algorithm (GA) [22], ant colony optimization (ACO) [12], particle swarm optimization (PSO) [24], DE [37], the quantum-inspired evolutionary algorithm (QEA) [21], harmony search (HS)[46], cuckoo search (CS) [44], teach-learning-based optimization (TLBO) [32], krill herd (KH) [18], the backtracking search optimization algorithm (BSA) [8], etc.

* Corresponding authors.
E-mail addresses: zhangkai@syuct.edu.cn (K. Zhang), jchsong@mail.neu.edu.cn (J. Song).

https://doi.org/10.1016/j.ins.2018.09.004
0020-0255/

In stochastic algorithms (i.e., meta-heuristics), most of them are population-based search algorithms. ACO, inspired by the foraging behavior of ants, has been proposed to solve combinatorial optimization problems [12]. To tackle continuous optimization problems, an extension of ACO for continuous domains (ACOR) was proposed [36]. QEA adopts the concept and principles of quantum computing, and it uses a Q-gate as a variation operator to lead the population toward convergence [21]. To improve the performance of the quantum algorithm, an adaptive quantum inspired genetic algorithm (aQIGA) was proposed in [5]. It uses adaptive control parameters to guide the distribution toward the better regions.

PSO has attracted much attention due to its ease of implementation and intelligent search capabilities. It is inspired by the social foraging behaviors of bird flocking and fish schooling. Unlike other EAs, PSO has two prominent characteristics. First, it does not use the law of survival of the fittest, and all particles are retained as members of the population through the evolutionary process. Second, it has no genetic operators such as mutation or crossover [17]. However, PSO is susceptible to falling into a local optimum solution in complex optimization problems. To improve PSO's performance, numerous significant variants of PSO have been proposed, which can be divided into three main categories: first, the parameters are introduced and modified, such as the inertia weight [33] and the linearly decreasing inertia weight [34]; second, an introduction of new topology structures or techniques is proposed, including the dynamic neighborhood learning strategy [27], human learning strategies [38], and simulated annealing [35]; and third, hybridization with other algorithms is adopted, such as HS [46], DE [14], and GA [19]. Liang et al. [26] presented a comprehensive learning PSO (CLPSO). Unlike PSO learning from the global best experience among the swarm, CLPSO learns from other particles' historical best experience. It enlarges the swarm's diversity to avoid premature convergence to improve PSO's performance on multimodal problems.

All the above mentioned EAs are nature-inspired population-based algorithms; however, they have some drawbacks, such as sensitivity to parameter settings. As the selection of parameters can significantly influence the performance of the algorithm, TLBO does not use algorithm-specific parameters [32]. Enhancement is accomplished either by introducing an elitism concept for TLBO (ETLBO) [31], or by utilizing a dynamic group strategy in TLBO (DGSTLBO) [47]. In DGSTLBO, learners can be regrouped dynamically and exchange information with each other. It protects the diversity of the population and avoids falling into the local optima to enhance TLBO's performance in addressing complex problems with multiple local optima.

BSA is a population-based stochastic search method with genetic operators [8]. It does not require algorithm-specific parameters, enabling an easy implementation and providing the flexibility to handle different complex optimization problems [6]. Duan and Luo [13] proposed an adaptive BSA with a varied mutation and crossover rate to optimize the mass and dimension of induction magnetometers. Wang et al. [41] hybridized BSA with DE to enhance the convergence of BSA. Chen et al. [7] developed the learning BSA (LBSA) by incorporating TLBO's learning idea in BSA and modifying the mutation process to improve the diversity of the population and the exploration capability of algorithm.

DE has a simple algorithmic structure, and it is a highly powerful and versatile evolutionary optimizer for complex search spaces. Hence, it is very popular and has been widely applied to address optimization problems in diverse fields [9]. Although DE shows great potential to solve many optimization problems, DE, similar to other EAs, also has some drawbacks, such as sensitivity to parameters, selection of mutation and crossover strategies [8], level of difficulty for solving functions that are not linearly separable [10], etc.

To enhance DE's performance, improvement studies of DE are conducted, which can be divided into two groups. The first group aims to modify the structures of DE. Brest et al. [3] proposed a novel approach to self-adaptively determine the control parameters for DE (jDE). When solving different computational problems, this method will adaptively adjust the control parameters of DE to obtain a better solution. The self-adaptive method is very efficient and easy to implement. Later, Qin et al. [30] presented a self-adaptive DE (SaDE), in which the mutation strategies and control parameters are self-adaptively updated in terms of previous experiences. Zhang and Sanderson [45] proposed a new DE variant that incorporates parameter adaptation and a new mutation strategy into DE to improve its performance and increase the convergence speed. Guo and Yang [20] proposed an eigenvector-based crossover operator to enhance DE's performance, which can also be applied to other DE versions. Ahmadianfar et al. [1] proposed the enhanced DE by utilizing two new mutation operators to balance exploration and exploitation capabilities, and applied it to optimize multi-reservoir operation problems.

The second group of improvement studies of DE focuses on integrating an extra component into DE. In this group, the modified versions of DE can be subdivided into two classes. In the first class, some studies consider a hybrid evolutionary framework of DE with other meta-heuristics, such as PSO [14], GA [40], HS [11] and so on. In the second class, in order to enhance DE's local exploitation capability, some limited studies are aimed at combining DE with local search methods. A crossover based on a local search scheme, the fittest individual refinement [28], was used for exploration around the best individual to help accelerate DE on high-dimensional functions. Afterwards, Noman and Iba [29] combined DE with an adaptive local search technique. It uses a hill-climbing heuristic to adaptively adjust the search length, which has been proven to be very promising. Tirronen et al. [39] presented a hybridization of DE with a scale factor local search algorithm to guarantee a high quality solution. Xie et al. [42] presented a diversity-maintained gradient-based DE (DMGBDE), which embeds a gradient local search method into DE to improve its local exploitation abilities, and introduced a diversity-maintained mutation operator to avoid premature convergence.

As mentioned above, DE has become popular due to its ease of implementation and has shown a promising performance on many global optimization problems. Both the adaptive DE with the parameter control and the hybridization of DE with a local search technique have shown faster and more reliable convergence performance than the basic DE.

To improve DE's performance on global optimization problems, we propose a novel algorithm, ADELI, which is based on DE by incorporating an adaptive local search scheme with Lagrange interpolation. First, Lagrange interpolation is introduced

into DE with a self-adaptive control parameter, and it performs local search in the neighborhood of the best individual. This local search scheme is used to enhance the exploitation capability to accelerate the convergence speed of DE. Second, an adaptive argument strategy is presented to adaptively determine whether to use Lagrange interpolation for the local search in one generation. This mechanism is used to improve the efficiency of ADELI.

In this paper, 30 benchmark functions from the CEC 2014 sets and an application of path generation are used to examine the feasibility of ADELI. The results are compared with those generated by ten widely used algorithms including ACOR [36], QEA [21], aQIGA [5], CLPSO [26], jDE [3], SaDE [30], ETLBO [31], DGSTLBO [47], BSA [8], and LBSA [7].

The rest of this paper is organized as follows. Section 2 briefly describes the procedure of the basic DE. Section 3 introduces Lagrange interpolation. The proposed ADELI is elaborated in Section 4, with detailed explanations on local search with Lagrange interpolation and the adaptive argument strategy. Numerical experiments are conducted in Section 5, and the application for path generation problems is optimized in Section 6. Finally, conclusions are summarized in Section 7.

2. DE

DE is a highly powerful and efficient evolutionary algorithm for solving optimization problems; it contains four operations: initialization, mutation, crossover, and selection, which are described below. Further details can be found in [37].

2.1. Initialization

The population of DE consists of N numbers of D-dimensional parameter vectors, i.e., $x_i = [x_{i,1}, x_{i,2}, ..., x_{i,D}], i = 1, 2, ..., N$. The parameters of the i th individual of the initial population are generated according to:

$$x_{i,j} = (\mathrm{x}_j^{\mathrm{up}} - \mathrm{x}_j^{low})) * \mathrm{rand}(0, 1) + \mathrm{x}_j^{low}, \mathrm{j} = 1, ..., \mathrm{D} \tag{1}$$

where $\mathrm{x}_j^{\mathrm{up}}$ and x_j^{low} are the upper and lower boundaries of individuals in the j th-dimensional search space, respectively, and $rand(0, 1)$ is a uniformly distributed random number in the range $(0, 1)$.

2.2. Mutation

After the initialization, the mutation procedure is employed. For each target vector x_i, $i = 1, 2, ..., N$, a mutant vector $v_{\mathrm{i}}^{\mathrm{child}}$ is generated by a mutation procedure. The mutation strategy is given as follows:

$$v_{\mathrm{i}}^{\mathrm{child}} = x_{ra} + F(x_{rb} - x_{rc}) \tag{2}$$

where ra, rb and rc are randomly chosen from $\{1, 2, ..., N\}$, which are different from each other. F is a user-specified scaling factor in the range $(0, 2)$.

If a mutation vector exceeds the upper and lower boundaries, it will be initialized by using Eq. (1).

2.3. Crossover

To increase the population diversity, the crossover operation is implemented to yield a trial vector $\mathrm{u}_i^{\mathrm{child}} = \{\mathrm{u}_{i,1}^{\mathrm{child}}, ..., \mathrm{u}_{i,D}^{\mathrm{child}}\}$. The crossover scheme of DE can be mathematically written as follows:

$$\mathrm{u}_{i,j}^{\mathrm{child}} = \begin{cases} v_{\mathrm{i,j}}^{\mathrm{child}} & if(\mathrm{rand}(0, 1) < \mathrm{CR}) \mathrm{or} (\mathrm{j} = \mathrm{j}_{\mathrm{rand}}) \\ x_{\mathrm{i,j}} & \text{otherwise} \end{cases} \quad \mathrm{j} = 1, 2, ..., D \tag{3}$$

where j_{rand} is randomly chosen from $\{1, 2, ..., D\}$ to ensure that $\mathrm{u}_i^{\mathrm{child}}$ inherits at least one element from $v_{\mathrm{i}}^{\mathrm{child}}$, and CR is the crossover rate defined by users within the range of $[0, 1]$.

2.4. Selection

After crossover, the trial vector $\mathrm{u}_{\mathrm{i}}^{\mathrm{child}}$ is calculated by the objective function. Then, the greedy selection mechanism is implemented to yield the population that can survive in the next generation in terms of the fitness value. If the fitness value of each trial vector $\mathrm{u}_{\mathrm{i}}^{\mathrm{child}}$ is better than the corresponding target vector x_{i}, then the trial vector $\mathrm{u}_{\mathrm{i}}^{\mathrm{child}}$ will survive into the next generation; otherwise, the target vector x_{i} will survive. The selection procedure can be mathematically expressed as Eq. (4).

$$\mathrm{x}_i^{\mathrm{child}} = \begin{cases} \mathrm{u}_{\mathrm{i}}^{\mathrm{child}} & if(\mathrm{f}(\mathrm{u}_{\mathrm{i}}^{\mathrm{child}}) < f(x_{\mathrm{i}})) \\ x_{\mathrm{i}} & \text{otherwise} \end{cases} \tag{4}$$

The pseudocode for DE is given in Algorithm 1.

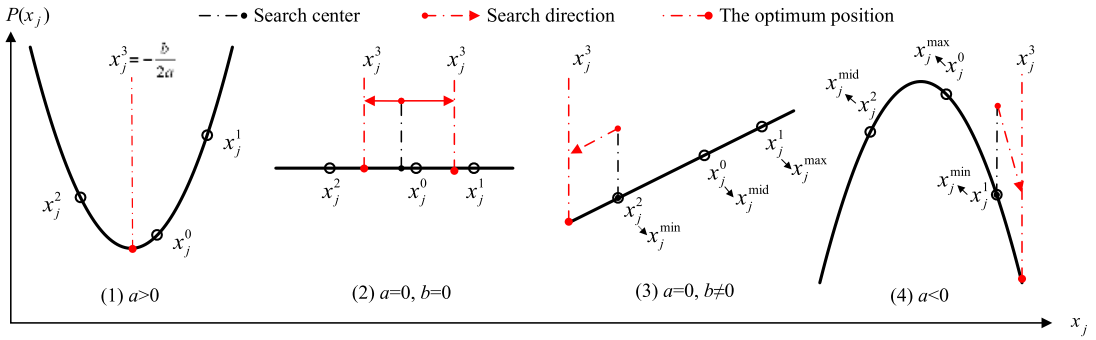

Fig. 1. Solutions of Lagrange interpolation in the j-th-dimensional space for $I \neq 0$.

3. Lagrange interpolation

Lagrange interpolation is a method to obtain a polynomial which can pass through all the given points accurately, rather than approximately. In the j th-dimensional search space, the Lagrange interpolation polynomial $P(x_j)$ of degree d passes through $d+1$ different given points $(x_j^0, f(x^0))$, $(x_j^1, f(x^1))$, ...,$(x_j^d, f(x^d))$, $j = 1, \ldots, D$, where x_j^k is the j th parameter in vector $x^k = \{x_1^k, x_2^k, ..., x_D^k\}$, $k = 0, \ldots, d$, and $f(x^k)$ is the corresponding fitness value of x^k, i.e., $f(x^k) = f([x_1^k, x_2^k, ..., x_D^k])$. It satisfies $P(x_j^k) = f(x^k)$, as shown in Eq. (5):

$$P(x_j) = \sum_{k=0}^{d} \left[\left(\prod_{\substack{m=0 \\ m \neq k}}^{d} \frac{x_j - x_j^m}{x_j^k - x_j^m} \right) f(x^k) \right], j = 1, ..., D \tag{5}$$

$P(x_j)$ is the interpolation polynomial of fitness $f(x)$. When degree $d = 2$, i.e., $P(x_j)$ passes through three points $(x_j^0, f(x^0))$, $(x_j^1, f(x^1))$ and $(x_j^2, f(x^2))$, it can be written as Eq. (6):

$$P(x_j) = \frac{(x_j - x_j^1)(x_j - x_j^2)}{(x_j^0 - x_j^1)(x_j^0 - x_j^2)} f(x^0) + \frac{(x_j - x_j^0)(x_j - x_j^2)}{(x_j^1 - x_j^0)(x_j^1 - x_j^2)} f(x^1) + \frac{(x_j - x_j^0)(x_j - x_j^1)}{(x_j^2 - x_j^0)(x_j^2 - x_j^1)} f(x^2) \quad j = 1, ..., D \tag{6}$$

Eq. (6) can be rewritten as:

$$P(x_j) = a(x_j)^2 + bx_j + c, j = 1, ..., D \tag{7}$$

where:

$$a = \frac{1}{I}\left[\left(x_j^2 - x_j^1\right) f(x^0) + \left(x_j^0 - x_j^2\right) f(x^1) + \left(x_j^1 - x_j^0\right) f(x^2) \right] \tag{8}$$

$$b = \frac{1}{I}\left[\left((x_j^1)^2 - (x_j^2)^2 \right) f(x^0) + \left((x_j^2)^2 - (x_j^0)^2 \right) f(x^1) + \left((x_j^0)^2 - (x_j^1)^2 \right) f(x^2) \right] \tag{9}$$

$$c = \frac{1}{I}\left[\left(x_j^2 - x_j^1\right) x_j^1 x_j^2 f(x^0) + \left(x_j^0 - x_j^2\right) x_j^0 x_j^2 f(x^1) + \left(x_j^1 - x_j^0\right) x_j^0 x_j^1 f(x^2) \right] \tag{10}$$

$$I = \left(x_j^0 - x_j^1\right)\left(x_j^1 - x_j^2\right)\left(x_j^2 - x_j^0\right) \tag{11}$$

The value of I decides the solution of the optimum position x_j^3. When $I \neq 0$, i.e., three points are distinct, as shown in Fig. 1, there are four cases:

- If $a > 0$, the parabola passes through three points and opens upwards; the minimal point of the parabola is the solution, i.e., $x_j^3 = -b/2a$.
- If $a = 0$, $b = 0$, three points arrange in a horizontal line, i.e., $f(x^0) = f(x^1) = f(x^2)$; x_j^3 can be randomly chosen between three points.
- If $a = 0$, $b \neq 0$, three points arrange in a non-horizontal straight line; x_j^3 can be randomly chosen at the neighbor of the minimum point among three points and can search along the decreasing direction of the line.
- If $a < 0$, the parabola opens downwards and there is no minimal point; x_j^3 can be solved as the case of $a = 0$, $b \neq 0$.

The solution of x_j^3 can be summarized as Eq. (12):

$$x_j^3 = \begin{cases} -\frac{b}{2a} & \text{if } a > 0 \quad (a) \\ \frac{1}{2}(x_j^1 + x_j^2) + (\text{rand} - 0.5)(x_j^1 - x_j^2) & \text{else if}(a = 0) \text{ and } (b = 0) \quad (b) \\ x_j^{\min} + \text{rand} * c_3 * (x_j^{\min} - x_j^{\text{mid}}) + \text{rand} * c_4 * (x_j^{\min} - x_j^{\max}) & \text{otherwise} \quad (c) \end{cases} \quad j = 1, \ldots, D \tag{12}$$

where x_j^0, x_j^1 and x_j^2 are three given points in the j th-dimension; $x_j^{\max}, x_j^{\text{mid}}$ and $x_j^{\min}$ are the maximum, middle, and minimum fitness values among the three given points, respectively; *rand* is a random number uniformly distributed in the range (0, 1) and c_3 and c_4 are two real numbers in the range (0, 1); here, $c_3 = 0.25$ and $c_4 = 0.25$.

When $I = 0$, i.e., at least two points are the same, the solution of x_j^3 can be solved as the case of $a = 0$, $b \neq 0$ using Eq. (c) in Eq. (12). Algorithm 2 shows the pseudocode of the Lagrange interpolation method.

4. ADELI

Although DE is a simple and powerful evolutionary algorithm for many optimization problems [30], it still has problems of premature convergence and a slow convergence rate for solving some complex optimization problems [15]. As Lagrange interpolation can enhance an algorithm's exploitation capability and help accelerate convergence [23], it is introduced to DE by using a novel adaptive argument strategy. Moreover, in order to solve complex optimization problems, a self-adaptive method for the control parameters is introduced for the parameter settings in DE. The new version of DE includes three main elements: local search with Lagrange interpolation; self-adapting control parameter settings in DE; and adaptive argument strategy.

4.1. Local search with Lagrange interpolation

The main aim of introducing LSLI is to enhance the exploitation capability of DE for local search at the neighbor of the individual with the best fitness (x^{best}). As $f(x^k) = f([x_1^k, x_2^k, \ldots, x_D^k])$, the fitness value of an individual is possibly influenced by all D parameters. The current best individual is discovered in the region nearby the global optimum. However, in some dimensions, a better fitness value may wait for further exploitation [26]. Hence, we implement LSLI for the local search in each dimension nearby the neighbor of the best individual to enhance the local search and convergence ability.

In each dimension of the search space, the initialization parameters of three temporary individuals x^1, x^2, and x^3 are assigned by the best individual x^{best}. Then, we empirically develop Eqs. (13) and (14) to randomly reassign the parameters x_j^1 and x_j^2 of the two temporary individuals x^1 and x^2 in each dimension.

$$x_j^1 = x_j^{\text{best}} + R_j \tag{13}$$

$$x_j^2 = x_j^{\text{best}} - R_j \tag{14}$$

where x_j^{best} is the parameter of the best individual in the j th-dimension, and R_j is the search radius of x^{best} in the j th-dimension; it can be adaptively adjusted as Eq. (15) according to the distribution of the current population.

$$R_j = \frac{1}{N}\left(\max\left(X_j\right) - \min\left(X_j\right)\right)\text{rand}(0, 1) \tag{15}$$

where X_j are the parameters of the population in the j th-dimension, N is the size of the population, and rand(0, 1) is a uniformly distributed random number in the range (0, 1).

In the j th-dimensional search space, the parameters x_j^1 and x_j^2 are on the opposite sides of the best individual's current position x_j^{best}. The parameters x_j^{best}, x_j^1 and x_j^2, treated as three points in the j th-dimensional search space, can perform Lagrange interpolation to obtain the parameter x_j^3 of the individual x^3 who may yield a better fitness value. If the fitness value $f(x^1)$, $f(x^2)$ or $f(x^3)$ is better than $f(x^{best})$, then the information of the best individual x^{best} will be updated. In each dimension, LSLI consumes 3 *FEs* for calculating x^1, x^2, and x^3. Hence, it consumes $3*D$ *FEs* in total when LSLI is performed one time.

To provide a better understanding, the pseudocode of LSLI is presented in Algorithm 3.

4.2. Self-adapting control parameters selection in DE

The performance of DE is significantly influenced by the control parameters, i.e., the scaling factor F and the crossover rate CR [30]. It is a difficult task to choose the suitable control parameters for optimizing increasingly complex real-world problems. In this paper, we introduce a simple and self-adaptive method, proposed by Brest et al. [3], to self-adaptively adjust the control parameters F and CR, which are given in Eqs. (16) and (17):

$$F_{i,G+1} = \begin{cases} F^{low} + rand(0, 1) * F^{up} & rand(0, 1) < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases} \tag{16}$$

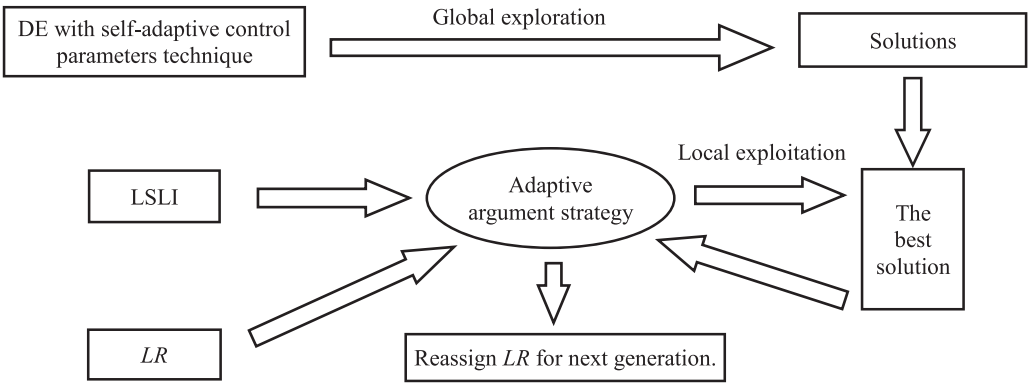

Fig. 2. The argument scheme of ADELI in one generation.

$$CR_{i,G+1} = \begin{cases} rand(0,1) & rand(0,1) < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \tag{17}$$

where F^{low} and F^{up} are the minimum and maximum of the control parameter F, respectively. τ_1 and τ_2 are probabilities to adjust the control parameters F and CR, respectively. G is the index of the generation.

4.3. Adaptive argument strategy

In ADELI, we insert LSLI into DE using an adaptive argument strategy, which is shown in Fig. 2. The main idea of ADELI is implemented in the following steps. First, DE with the self-adaptive control parameters method globally explores a problem's solution quickly in the entire search space. Then, the scheme decides whether to use LSLI according to the LSLI rate, *LR*, which is assigned by utilizing the feedback from the performance of LSLI in previous generation. Finally, the scheme adaptively determines whether to use LSLI for local exploitation near the best solution discovered by DE and to update *LR*.

LR decides the probability of using LSLI in this generation. If the uniform random number is smaller than *LR*, LSLI will be implemented around the best individual; otherwise LSLI will not be implemented in this generation.

After LSLI and before the mutation procedure, if the offspring's fitness value $f(x_{son}^{best})$ is better than the parent's best fitness value $f(x^{best})$, the offspring x_{son}^{best} will survive and perform the mutation operation, and *LR* will be adaptively reassigned with a larger value, which means a larger probability exists to use LSLI in the next generation; otherwise, the individual with the best fitness in the parents will remain and perform the mutation operation, and *LR* will be reassigned with a smaller value, which means a smaller probability exists to use LSLI the in next generation. *LR* is adaptively reassigned as Eq. (18).

$$LR_{G+1} = \begin{cases} \min(LR_G + 1, LR^{max}) & \text{if } (f(x_{son}^{best}) < f(x^{best})) \\ \max(LR_G - 1, LR^{min}) & \text{otherwise} \end{cases} \tag{18}$$

where $f(x^{best})$ and $f(x_{son}^{best})$ are the fitness values of the best parent and offspring individuals, respectively, when LSLI is implemented. LR^{min} and LR^{max} are two user-specified constants in the range [0, 1], which control the minimum and maximum of *LR*, respectively. G is the index of the generation.

Note that ADELI only performs LSLI in the neighborhood of the best individual. Hence, the individuals should be ranked after the initialization and the selection procedures; then, the best individual should be memorized and updated in terms of the fitness value. The termination criteria for ADELI are *FEs*. If LSLI is implemented in a generation, it would need $3*D$ additional *FEs*; otherwise it needs N *FEs* in this generation. Algorithm 4 shows the pseudocode of ADELI.

5. Numerical experiments

5.1. Benchmark functions

In this experimentation, 30 benchmark functions in the CEC 2014 sets [25] are used to test the performance of the ADELI algorithm. The definitions of the CEC 2014 sets can be found in [25]. These 30 benchmark functions can be classified into four categories: unimodal functions (F_1–F_3), simple multimodal functions (F_4–F_{16}), hybrid functions (F_{17}–F_{22}), and composition functions (F_{23}–F_{30}). All benchmark functions are shifted to the shifted global optimum $\boldsymbol{o}$, which is randomly distributed in the range $[-80, 80]^D$, where $\boldsymbol{o}_i = [o_{i1}, o_{i2}, ..., o_{iD}]^T$, and $\boldsymbol{M}$ is the rotation matrix provided in [25]. The search ranges of all benchmarks are the same, within the range of $[-100, 100]^D$.

Table 1
Parameter settings of all involved algorithms.

No.	Algorithm	Year	Parameter setting	Reference
1.	ACOR	2008	ξ=0.85 (speed of convergence), $q=10^{-4}$ (locality of the search process), $k=50$ (archive size)	[36]
2.	QEA	2002	$\Delta\theta=0.01\pi$ (magnitude of rotation angle), migration period = 100	[21]
3.	aQIGA	2014	$Pc=0.5$ (crossover rate), $\sigma=1$ (standard deviation)	[5]
4.	CLPSO	2006	$w=0.9-0.4$, $c_1=1.49445$, $m_r=7$ (refreshing gap), $Pc_i=0.05+0.45[\exp(10(i-1)/(N-1)-1)]/(\exp(10)-1)$(learning probability)	[26]
5.	jDE	2006	$CR=0.9$, $F=0.5$, $F^{low}=0.1$, $F^{up}=0.9$, $\tau_1=0.1$, $\tau_2=0.1$	[3]
6.	SaDE	2009	$F\sim N(0.5,0.3)$, $CR_0=0.5$, $CR\sim N(CRm, 0.1)$, $LP=50$ (learning period)	[30]
7.	ETLBO	2012	elite size =2	[31]
8.	DGSTLBO	2014	$m_g=5$ (grouping size), $g=5$, $Pc=0.5(G-1)/(\max G-1)$	[47]
9.	BSA	2013	–	[8]
10.	LBSA	2017	–	[7]
11.	ADELI		$CR=0.9$, $F=0.5$, $F^{low}=0.1$, $F^{up}=0.9$, $\tau_1=0.1$, $\tau_2=0.1$, $LR^{min}=0.1$, $LR^{max}=0.9$	

5.2. Experimental settings

To evaluate the efficiency of ADELI, ten evolutionary algorithms including ACOR [36], QEA [21], aQIGA [5], CLPSO [26], two DE versions, (jDE [3] and SaDE [30]), two TLBO versions (ETLBO [31] and DGSTLBO [47]), and two BSA versions (BSA [8] and LBSA [7]) are evaluated to compare with the performance of ADELI.

For a fair comparison, all experiments are carried out on an Intel (R) i7 4850-HQ processor at 2.30 GHz with MATLAB R2016a. Each algorithm is independently implemented 30 times. In each experiment, the population size is set at 50 in the problems of 30 dimensions (30D) and 50 dimensions (50D). The termination criterion for all algorithms is the number of fitness evaluations (*FEs*); hence, the maximum *FEs* is set at 250,000. Note that for ACOR, QEA, aQIGA, CLPSO, and DE versions, the expenditure of *FEs* is N in one generation. For TLBO versions, the expenditure of *FEs* is $2N$ in one generation. For ADELI, if a generation performed LSLI one time, it would need $3D$ additional *FEs* in that generation, i.e., $(3D+N)$ *FEs*; otherwise, the generation would need N *FEs*. In this numerical experiment, the fitness value is the error between the optimal solution obtained by the algorithm and the real optimum.

The parameter settings of these algorithms are listed in Table 1, which are recommended in the original literature. Similar to most DE variants, ADELI requires setting the control parameters, i.e., the scaling factor (F) and the crossover rate (CR); however, it is difficult to choose suitable values. Hence, we use the same selection strategy as jDE [3]. The values of F^{low}, F^{up}, τ_1, and τ_2 are the same as jDE and given in Table 1.

In addition, LR is an important parameter of ADELI that balances exploration and exploitation between DE and LSLI. Hence, an adaptive argument strategy is used to determine LR by LR^{min} and LR^{max}. If LSLI is unsuccessful, LR is set to LR^{min}, which means that LSLI cannot achieve a better solution than the best individual in this generation, and LSLI may not be suitable for this region. In this paper, we set LR^{min} with a small value of 0.1. Conversely, LR will be reassigned by using a large probability LR^{max} when LSLI obtains a better solution. In this study, we test four ADELIs with different LR^{max} {0.6, 0.7, 0.8, 0.9} and compare them with jDE. The mean error, mean time, and mean LSLI ratio obtained from 30 runs for the 30D functions in the CEC 2014 sets are shown in Table 2.

To observe the utilization rate of LSLI, we introduce the LSLI ratio, which is defined by

$$\text{LSLI } ratio = \frac{\text{FEs}^{LSLI}}{\text{Max FEs}} \tag{19}$$

where Max*FEs* is the maximum *FEs*, and FEs^{LSLI} is the total expenditure of *FEs* of LSLI.

From the results in Table 2, we can observe that the adaptive argument mechanism of ADELI can work well by using different LSLI ratios on different problems. When LR^{max} is set to 0.9, ADELI performs the best, achieving the best mean error on 11 functions. In addition, all ADELIs spends less CPU time than jDE on most functions, and the higher LR^{max} it uses, the less time it takes. Considering two main factors, i.e., computational cost and solution accuracy, comprehensively, we set LR^{max} to 0.9 in our numerical experiments.

5.3. Comparison of the solution accuracy and efficiency

5.3.1. Results for the 30D problems

Table 3 shows the mean time of ADELI compared with the other ten algorithms for the 30D benchmark functions over 30 independent runs. We can observe that BSA, ADELI, LBSA, and jDE are the top four algorithms which spend the least time on most functions.

The mean, standard deviation and best solutions of eleven algorithms on thirty test functions with 30 dimensions (30D) are shown in Table 4. The best results among the eleven algorithms are shown in boldface. Moreover, the rank in terms of the mean for each function, and its average rank of eleven algorithms are presented in Table 4.

Table 2
The mean error, mean time and mean LSLI ratio on thirty 30D test functions with different LR^{max} (the boldface is the best mean error).

	jDE			$LR^{max}=0.6$			$LR^{max}=0.7$			$LR^{max}=0.8$			$LR^{max}=0.9$		
Func.	Mean error	Mean time	Mean LSLI ratio	Mean error	Mean time	Mean LSLI ratio	Mean error	Mean time	Mean LSLI ratio	Mean error	Mean time	Mean LSLI ratio	Mean error	Mean time	Mean LSLI ratio
F1	3.37E+05	3.26	0%	1.19E+05	3.42	51%	1.03E+05	2.84	55%	1.46E+05	2.87	58%	8.14E+04	2.78	62%
F2	1.99E-14	2.84	0%	0.00E+00	2.41	41%	8.38E-23	2.29	48%	0.00E+00	2.29	49%	1.25E-21	2.21	53%
F3	4.55E-14	2.80	0%	1.12E-20	2.27	49%	4.26E-20	2.27	53%	2.58E-17	2.15	57%	8.53E-16	2.12	61%
F4	2.49E+01	2.80	0%	4.50E-05	2.26	52%	4.09E-05	2.22	55%	2.83E-03	2.16	59%	7.51E-05	2.13	62%
F5	2.04E+01	3.17	0%	2.00E+01	3.06	18%	2.00E+01	3.03	18%	2.00E+01	3.03	18%	2.00E+01	3.02	18%
F6	1.48E+01	31.84	0%	3.18E-01	31.70	34%	4.80E-05	31.57	38%	7.05E-01	31.64	36%	9.56E-01	31.62	38%
F7	1.71E-13	3.21	0%	4.90E-02	3.01	19%	6.88E-03	3.04	20%	1.03E-02	3.00	19%	8.85E-03	3.02	20%
F8	1.14E-14	2.72	0%	0.00E+00	2.56	20%	0.00E+00	2.65	20%	0.00E+00	2.63	21%	0.00E+00	2.65	21%
F9	5.82E+01	3.11	0%	6.61E+01	3.07	18%	7.88E+01	3.05	19%	7.62E+01	3.06	18%	6.53E+01	3.05	19%
F10	1.69E-01	3.03	0%	1.19E+01	3.02	20%	1.32E+01	3.00	20%	1.24E+01	2.99	21%	1.13E+01	2.92	20%
F11	2.88E+03	3.45	0%	3.12E+03	3.29	19%	3.62E+03	3.35	17%	3.01E+03	3.29	19%	2.76E+03	3.31	19%
F12	4.96E-01	12.31	0%	3.59E-01	12.41	16%	3.45E-01	12.40	17%	3.86E-01	12.31	16%	2.76E-01	12.43	16%
F13	3.10E-01	2.79	0%	2.38E-01	2.83	16%	2.56E-01	2.82	15%	2.44E-01	2.84	16%	2.36E-01	2.83	16%
F14	2.95E-01	2.77	0%	2.42E-01	2.68	15%	2.23E-01	2.71	15%	2.26E-01	2.74	15%	2.53E-01	2.70	16%
F15	5.81E+00	3.12	0%	6.31E+00	2.91	29%	5.34E+00	2.83	33%	5.02E+00	2.92	34%	3.77E+00	2.87	29%
F16	1.06E+01	3.18	0%	1.07E+01	2.99	22%	1.07E+01	3.00	23%	1.04E+01	3.01	22%	1.05E+01	2.97	23%
F17	3.16E+04	3.33	0%	4.20E+03	2.83	50%	1.61E+04	2.76	54%	9.81E+03	2.87	56%	6.47E+03	2.66	60%
F18	2.63E+02	2.93	0%	2.58E+01	2.80	20%	1.93E+01	2.79	20%	2.47E+01	2.80	20%	1.60E+01	2.85	22%
F19	1.41E+01	8.80	0%	3.97E+00	8.46	49%	3.96E+00	8.41	54%	4.26E+00	8.33	56%	3.79E+00	8.23	60%
F20	3.17E+01	3.01	0%	1.49E+01	2.79	28%	2.18E+01	2.82	28%	1.56E+01	2.82	30%	1.93E+01	2.83	29%
F21	3.66E+03	3.36	0%	1.11E+03	2.78	44%	1.43E+03	2.68	51%	1.62E+03	2.62	56%	2.61E+03	2.61	57%
F22	2.96E+02	4.13	0%	7.95E+01	4.02	23%	1.12E+02	3.95	24%	8.90E+01	3.97	24%	9.70E+01	3.93	24%
F23	3.14E+02	5.37	0%	3.15E+02	5.08	35%	3.15E+02	5.07	39%	3.15E+02	5.01	42%	3.15E+02	4.98	44%
F24	2.27E+02	4.17	0%	2.24E+02	4.11	18%	2.25E+02	4.08	21%	2.24E+02	4.08	20%	2.24E+02	4.12	20%
F25	2.00E+02	5.18	0%	2.03E+02	4.77	47%	2.03E+02	4.76	48%	2.04E+02	4.77	49%	2.04E+02	4.74	50%
F26	1.00E+02	35.05	0%	1.00E+02	35.34	16%	1.00E+02	35.29	16%	1.00E+02	35.39	16%	1.00E+02	35.43	16%
F27	5.65E+02	34.88	0%	3.81E+02	35.01	38%	4.01E+02	35.03	42%	3.61E+02	35.00	42%	3.81E+02	35.04	40%
F28	3.83E+02	6.45	0%	8.30E+02	6.33	27%	8.18E+02	6.32	24%	8.22E+02	6.32	24%	8.51E+02	6.20	35%
F29	2.15E+02	10.41	0%	1.09E+03	10.15	40%	1.06E+03	10.16	44%	1.20E+03	10.15	45%	9.88E+02	10.18	42%
F30	3.87E+02	5.60	0%	1.14E+03	5.34	42%	1.16E+03	5.38	40%	7.22E+02	5.35	41%	1.25E+03	5.39	38%

Table 3
The mean time, averaged over 30 independent runs, of thirteen algorithms on 30D test functions.

Func.	ACOR	QEA	aQIGA	CLPSO	jDE	SaDE	ETLBO	DGSTLBO	BSA	LBSA	ADELI	
F1	40.67	38.68	43.92	5.96	3.26	27.26	12.04	3.47		2.37	2.99	2.78
F2	39.37	37.42	41.51	4.65	2.84	28.88	10.96	2.81		1.73	2.39	2.21
F3	38.63	38.25	41.09	5.31	2.80	26.22	11.44	2.75		1.70	2.35	2.12
F4	38.60	36.57	41.20	5.39	2.80	25.62	10.89	2.77		1.72	2.36	2.13
F5	38.97	36.84	41.26	6.30	3.17	27.11	11.47	3.08		2.08	2.69	3.02
F6	68.31	65.82	70.69	34.90	31.84	55.73	55.31	32.31		30.98	31.55	31.62
F7	39.24	36.91	41.53	5.13	3.21	27.19	11.58	3.19		2.09	2.70	3.02
F8	38.61	36.47	41.13	4.97	2.72	25.84	10.75	2.63		1.62	2.24	2.65
F9	39.01	36.84	41.32	6.05	3.11	26.17	11.35	3.05		2.05	2.65	3.05
F10	38.90	36.85	41.26	5.75	3.03	26.30	11.09	2.92		1.95	2.57	2.92
F11	39.38	37.16	41.67	6.49	3.45	26.79	11.65	3.33		2.36	2.96	3.31
F12	48.45	46.12	50.82	15.42	12.31	35.83	24.98	12.27		11.18	11.78	12.43
F13	38.77	36.56	41.35	5.64	2.79	25.00	10.96	2.74		1.72	2.33	2.83
F14	38.60	36.52	41.33	5.58	2.77	25.05	10.83	2.72		1.69	2.32	2.70
F15	38.96	36.91	41.59	5.81	3.12	25.54	11.28	3.09		2.04	2.67	2.87
F16	39.29	36.89	41.45	6.22	3.18	26.44	11.21	3.06		2.13	2.69	2.97
F17	39.33	37.09	41.82	5.91	3.33	26.33	11.67	3.35		2.22	2.84	2.66
F18	38.81	36.86	41.28	5.56	2.93	25.86	10.99	2.89		1.84	2.45	2.85
F19	45.06	42.77	47.46	11.40	8.80	31.57	19.89	8.84		7.75	8.37	8.23
F20	39.18	36.84	41.53	5.83	3.01	25.88	11.10	3.04		1.95	2.58	2.83
F21	39.12	36.89	42.43	5.70	3.36	26.20	11.44	3.18		2.04	2.65	2.61
F22	40.29	38.19	42.80	6.92	4.13	27.45	12.99	4.10		3.05	3.67	3.93
F23	42.04	39.28	44.17	7.36	5.37	27.47	15.10	5.41		4.27	4.89	4.98
F24	40.68	38.21	42.85	6.70	4.17	26.41	13.30	4.19		3.10	3.71	4.12
F25	41.83	39.11	43.87	8.05	5.18	27.32	14.79	5.25		4.09	4.69	4.74
F26	72.75	69.62	74.64	37.88	35.05	57.63	59.63	35.37		34.01	34.61	35.43
F27	72.25	69.54	74.70	37.87	34.88	58.24	59.48	35.22		33.94	34.41	35.04
F28	42.92	40.56	45.33	9.33	6.45	28.83	17.00	6.70		5.54	6.14	6.20
F29	46.97	44.55	49.25	12.96	10.41	33.59	22.51	10.68		9.29	9.87	10.18
F30	42.36	39.60	44.41	8.21	5.60	28.36	15.48	5.77		4.51	5.14	5.39

In Table 4, it can be observed that ADELI outperforms the other ten algorithms, with respect to the mean, on fourteen functions (F1, F2, F3, F4, F5, F8, F12, F13, F15, F17, F18, F19, F20, and F22). jDE surpasses other algorithms on five functions (F10, F23, F28, F29, and F30). SaDE is superior to other algorithms on four functions (F6, F7, F21, and F27). LBSA performs the best among all algorithms on three functions (F9, F11, and F14). For function F16, aQIGA and LBSA have the best performance, obtaining the minimum mean. For function F24, ETLBO, and DGSTLBO perform better than the other algorithms in terms of the mean. For function F25, jDE and ETLBO have the least mean. For function F26, CLPSO, jDE, SaDE, ETLBO, BSA, LBSA and ADELI outperform other algorithms with respect to the mean. ADELI ranks in the first place on fifteen functions, jDE on seven functions, and SaDE and LBSA on five functions. Additionally, the average rank of ADELI for 30D problems is 2.23, which is the best value among all competitor algorithms. Hence, it can be concluded that ADELI performs the best, and achieves good solution accuracy on 30-dimensional global optimization problems.

5.3.2. Results for the 50D problems

Table 5 presents the mean, standard deviation, best solution, rank of mean, and its average rank on thirty benchmark functions in 50 dimensions. The best results are shown in boldface. From Table 5, it can be observed that ADELI surpasses all other ten algorithms in terms of the mean on functions F1, F2, F3, F4, F13, F15, F17, F18, F19, F20, F21, and F22. ETLBO performs the best among all algorithms on functions F5, F12, and F27. LBSA outperforms other algorithms on functions F8, F11, and F14. jDE is superior to other algorithms on functions F6 and F7. SaDE performs better than the other algorithms on functions F9 and F10. ACOR performs the best with the smallest mean on functions F29 and F30. aQIGA surpasses other algorithms on function F16. For functions F23 and F28, ETLBO and DGSTLBO outperform other algorithms. For functions F24 and F25, ETLBO, and DGSTLBO obtain the least mean. For function F26, BSA and ADELI perform the best with respect to the mean. ADELI ranks in the first place on thirteen functions, ETLBO on seven functions, LBSA on four functions, DGSTLBO on three functions, jDE and SaDE on two functions, and BSA on one function. Additionally, the average rank of ADELI for 50D problems is 2.93, which is the best value among all competitor algorithms. Therefore, it can be concluded that ADELI performs the best, and achieves good solution accuracy on 50-dimensional global optimization problems.

Considering the average ranks of 30D functions and 50D functions, the top four methods are ADELI (2.58), LBSA (3.40), jDE (3.84), and SaDE (4.30). Hence, as indicated in Tables 3–5, ADELI shows the best overall performance among all algorithms on 30D and 50D problems, considering the mean of the error values.

Table 4
Comparative results of eleven algorithms on 30D test functions (the bold is the best solution).

Func.	Al.	ACOR	QEA	aQIGA	CLPSO	jDE	SaDE	ETLBO	DGSTLBO	BSA	LBSA	ADELI
F1	Mean	1.90E+07	8.41E+07	5.70E+07	2.92E+07	3.37E+05	4.41E+05	3.33E+05	1.04E+07	5.12E+06	2.79E+05	**8.14E+04**
	Std.	8.11E+06	8.04E+07	3.13E+07	4.33E+06	2.98E+05	3.04E+05	2.02E+05	8.61E+06	4.10E+06	1.65E+05	**8.69E+04**
	Best	7.16E+06	2.24E+07	1.80E+07	2.14E+07	1.06E+05	1.23E+05	1.37E+05	3.15E+06	4.46E+05	7.85E+04	**3.72E+04**
	Rank	8	11	10	9	4	5	3	7	6	2	1
F2	Mean	1.87E+06	4.25E+09	1.31E+09	1.27E+03	1.99E-14	2.84E-15	1.19E+02	4.59E+06	8.34E-01	4.83E-14	**1.25E-21**
	Std.	3.55E+06	1.01E+09	1.86E+09	7.40E+02	1.37E-14	8.99E-15	1.56E+02	1.11E+07	1.48E+00	1.92E-14	**2.62E-21**
	Best	1.97E+05	2.77E+09	2.91E+08	2.76E+02	**0.00E+00**	**0.00E+00**	3.29E-01	1.49E+03	5.38E-02	2.84E-14	**0.00E+00**
	Rank	8	11	10	7	3	2	6	9	5	4	1
F3	Mean	1.05E+04	3.63E+04	1.99E+04	7.67E+02	4.55E-14	4.30E-12	2.14E+03	1.44E+01	5.74E-04	2.27E-13	**8.53E-16**
	Std.	1.81E+04	3.43E+04	1.67E+04	1.20E+03	2.40E-14	1.25E-11	1.02E+03	1.68E+01	9.44E-04	1.00E-13	**1.04E-15**
	Best	2.63E+02	3.33E+03	3.90E+03	5.76E+01	**0.00E+00**	**0.00E+00**	5.59E+02	6.71E-01	3.13E-05	1.14E-13	3.93E-20
	Rank	9	11	10	7	2	4	8	6	5	3	1
F4	Mean	1.23E+02	4.38E+02	2.69E+02	1.16E+02	2.49E+01	8.29E+01	7.53E+01	1.46E+02	9.83E+01	4.23E+01	**7.51E-05**
	Std.	3.09E+01	1.70E+02	5.31E+01	1.33E+01	1.74E+01	2.70E+01	5.88E+00	3.78E+01	2.96E+01	3.57E+01	**1.19E-04**
	Best	8.06E+01	1.94E+02	2.26E+02	9.56E+01	1.81E+01	6.73E+01	6.79E+01	8.79E+01	6.81E+01	1.73E-03	**2.79E-07**
	Rank	8	11	10	7	2	5	4	9	6	3	1
F5	Mean	2.09E+01	2.01E+01	2.01E+01	2.05E+01	2.04E+01	2.06E+01	2.09E+01	2.10E+01	2.04E+01	2.03E+01	**2.00E+01**
	Std.	8.81E-02	5.57E-02	6.69E-02	4.58E-02	3.56E-02	5.89E-02	7.24E-02	4.34E-02	1.56E-02	3.42E-02	**1.12E-03**
	Best	2.08E+01	**2.00E+01**	2.01E+01	2.04E+01	2.03E+01	2.05E+01	2.08E+01	2.09E+01	2.03E+01	2.03E+01	**2.00E+01**
	Rank	9	2	2	7	5	8	9	11	5	4	1
F6	Mean	1.67E+01	2.34E+01	1.84E+01	1.70E+01	1.48E+01	**2.61E-01**	1.59E+01	1.67E+01	1.62E+01	8.37E+00	9.56E-01
	Std.	2.03E+00	1.35E+00	2.53E+00	6.10E-01	3.97E+00	**4.92E-01**	2.75E+00	3.45E+00	9.69E-01	3.22E+00	1.45E+00
	Best	1.41E+01	2.15E+01	1.50E+01	1.60E+01	4.01E+00	**0.00E+00**	1.01E+01	1.23E+01	1.47E+01	5.25E+00	2.07E-05
	Rank	7	11	10	9	4	1	5	7	6	3	2
F7	Mean	5.59E-01	5.41E+01	1.44E+01	3.92E-03	1.71E-13	**0.00E+00**	2.75E-02	1.01E+00	4.19E-03	5.41E-03	8.85E-03
	Std.	2.90E-01	1.88E+01	6.09E+00	1.61E-03	8.04E-14	**0.00E+00**	2.23E-02	1.50E+00	1.32E-02	8.26E-03	1.60E-02
	Best	3.80E-01	2.78E+01	5.59E+00	1.37E-03	1.14E-13	**0.00E+00**	1.02E-12	1.20E-01	6.62E-07	1.14E-13	**0.00E+00**
	Rank	8	11	10	3	2	1	7	9	4	5	6
F8	Mean	1.09E+02	7.52E+01	4.66E+01	5.86E-01	1.14E-14	1.99E-01	8.00E+01	7.67E+01	2.93E+00	1.14E-13	**0.00E+00**
	Std.	1.65E+01	6.11E+00	1.27E+01	6.20E-01	3.60E-14	4.20E-01	1.88E+01	2.45E+01	1.46E+00	**0.00E+00**	**0.00E+00**
	Best	9.15E+01	6.53E+01	3.58E+01	2.65E-04	**0.00E+00**	**0.00E+00**	4.48E+01	3.48E+01	1.14E+00	1.14E-13	**0.00E+00**
	Rank	11	8	7	5	2	4	10	9	6	3	1
F9	Mean	1.57E+02	1.49E+02	9.59E+01	7.79E+01	5.82E+01	7.70E+01	8.00E+01	9.84E+01	5.95E+01	**4.32E+01**	6.53E+01
	Std.	3.14E+01	3.50E+01	2.65E+01	1.49E+01	7.79E+00	1.68E+01	1.42E+01	3.08E+01	7.94E+00	**7.61E+00**	1.75E+01
	Best	1.21E+02	1.07E+02	6.48E+01	5.52E+01	4.78E+01	4.47E+01	5.87E+01	6.41E+01	4.56E+01	**2.70E+01**	5.07E+01
	Rank	11	10	8	6	2	5	7	9	3	1	4
F10	Mean	1.73E+03	1.64E+03	5.30E+02	2.50E+01	**1.69E-01**	2.56E+00	1.58E+03	2.40E+03	3.22E+01	1.08E+01	1.13E+01
	Std.	5.06E+02	2.91E+02	1.61E+02	5.21E+00	**5.12E-01**	4.09E+00	4.26E+02	4.71E+02	6.57E+00	5.07E+00	5.29E+00
	Best	1.13E+03	1.30E+03	3.19E+02	1.83E+01	**0.00E+00**	3.39E-02	9.52E+02	1.60E+03	2.15E+01	5.12E+00	7.53E+00
	Rank	10	9	7	5	1	2	8	11	6	3	4
F11	Mean	2.71E+03	3.46E+03	3.47E+03	3.24E+03	2.89E+03	4.11E+03	6.47E+03	3.40E+03	2.57E+03	**2.31E+03**	2.77E+03
	Std.	5.03E+02	1.10E+03	2.64E+02	**2.18E+02**	3.06E+02	4.80E+02	4.46E+02	5.45E+02	2.56E+02	3.17E+02	6.28E+02
	Best	2.08E+03	2.10E+03	3.01E+03	2.99E+03	2.51E+03	3.19E+03	5.52E+03	2.42E+03	2.12E+03	**1.87E+03**	2.09E+03
	Rank	3	8	9	6	5	10	11	7	2	1	4
F12	Mean	2.44E+00	3.37E-01	3.16E-01	5.42E-01	4.96E-01	1.05E+00	2.60E+00	2.75E+00	4.37E-01	4.18E-01	**2.76E-01**
	Std.	1.67E-01	1.83E-01	1.21E-01	8.53E-02	5.71E-02	1.24E-01	3.32E-01	2.62E-01	7.85E-02	**5.15E-02**	1.52E-01
	Best	2.27E+00	1.95E-01	1.96E-01	4.12E-01	3.72E-01	7.98E-01	1.93E+00	2.48E+00	3.05E-01	3.62E-01	**1.87E-01**
	Rank	9	3	2	7	6	8	10	11	5	4	1
F13	Mean	4.69E-01	6.32E-01	6.58E-01	4.18E-01	3.10E-01	3.09E-01	4.11E-01	4.71E-01	2.84E-01	2.90E-01	**2.36E-01**
	Std.	1.02E-01	1.37E-01	1.03E-01	7.14E-02	4.77E-02	**3.58E-02**	6.86E-02	1.13E-01	4.68E-02	4.34E-02	3.70E-02
	Best	3.67E-01	4.80E-01	4.87E-01	2.77E-01	2.58E-01	2.62E-01	2.97E-01	3.09E-01	2.06E-01	2.10E-01	**1.86E-01**
	Rank	8	10	11	7	5	4	6	9	2	3	1
F14	Mean	5.73E-01	8.54E+00	3.22E-01	3.44E-01	2.95E-01	2.81E-01	2.78E-01	2.88E-01	2.45E-01	**2.30E-01**	2.53E-01
	Std.	3.74E-01	7.14E+00	3.80E-02	3.38E-02	2.67E-02	**2.64E-02**	4.42E-02	4.92E-02	4.02E-02	2.69E-02	3.07E-02
	Best	2.68E-01	2.94E-01	2.82E-01	2.97E-01	2.56E-01	2.25E-01	**1.84E-01**	2.26E-01	2.03E-01	1.87E-01	2.17E-01
	Rank	10	11	8	9	7	5	4	6	2	1	3
F15	Mean	1.38E+02	6.52E+02	4.83E+02	1.09E+01	5.81E+00	9.71E+00	1.89E+01	3.75E+01	7.06E+00	5.93E+00	**3.77E+00**
	Std.	1.60E+02	8.19E+02	8.05E+02	9.63E-01	**8.14E-01**	1.28E+00	8.66E+00	2.19E+01	1.07E+00	1.22E+00	1.97E+00
	Best	3.38E+01	1.02E+02	2.71E+01	9.09E+00	4.24E+00	7.32E+00	9.75E+00	1.54E+01	5.16E+00	3.87E+00	**2.02E+00**
	Rank	9	11	10	6	2	5	7	8	4	3	1
F16	Mean	1.15E+01	1.19E+01	**1.02E+01**	1.11E+01	1.06E+01	1.15E+01	1.19E+01	1.11E+01	1.07E+01	**1.02E+01**	1.05E+01
	Std.	8.36E-01	7.91E-01	7.10E-01	2.31E-01	**1.90E-01**	2.07E-01	3.94E-01	6.62E-01	2.71E-01	4.55E-01	5.86E-01
	Best	1.03E+01	1.09E+01	9.48E+00	1.07E+01	1.03E+01	1.13E+01	1.14E+01	1.01E+01	1.01E+01	**9.37E+00**	9.91E+00
	Rank	8	10	1	6	4	8	10	6	5	1	3
F17	Mean	1.19E+06	9.35E+06	2.78E+06	2.07E+06	3.16E+04	1.33E+04	2.11E+05	1.67E+05	1.54E+05	4.42E+04	**6.48E+03**
	Std.	6.69E+05	8.22E+06	1.31E+06	6.62E+05	3.34E+04	6.50E+03	9.21E+04	2.13E+05	8.75E+04	3.70E+04	**2.53E+03**
	Best	4.49E+05	3.04E+06	1.65E+06	1.07E+06	2.50E+03	**1.73E+03**	8.02E+04	3.54E+04	4.92E+04	8.62E+03	3.50E+03
	Rank	8	11	10	9	3	2	7	6	5	4	1
F18	Mean	1.12E+04	1.47E+08	3.95E+07	4.92E+02	2.63E+02	6.89E+01	3.31E+03	8.71E+02	9.10E+02	1.14E+03	**1.60E+01**

(*continued on next page*)

Table 4 (*continued*)

	Std.	1.20E+04	9.14E+07	5.38E+07	9.30E+01	5.11E+02	3.43E+01	4.11E+03	1.02E+03	1.06E+03	1.30E+03	**4.29E+00**
	Best	7.54E+02	5.88E+07	4.34E+05	3.44E+02	1.35E+01	3.09E+01	1.64E+02	6.17E+01	4.35E+01	6.23E+01	**9.32E+00**
	Rank	9	11	10	4	3	2	8	5	6	7	1
F19	Mean	5.56E+01	1.35E+02	5.95E+01	1.01E+01	1.41E+01	5.55E+00	2.12E+01	2.71E+01	6.91E+00	6.31E+00	**3.79E+00**
	Std.	9.11E+01	7.73E+01	3.17E+01	1.02E+00	1.13E+00	6.59E-01	2.54E+01	2.86E+01	6.25E-01	1.01E+00	**5.33E-01**
	Best	1.27E+01	8.14E+01	2.24E+01	8.20E+00	1.19E+01	4.74E+00	6.72E+00	9.65E+00	6.06E+00	4.60E+00	**3.29E+00**
	Rank	9	11	10	5	6	2	7	8	4	3	1
F20	Mean	5.24E+02	5.05E+04	4.43E+04	5.96E+03	3.17E+01	4.31E+01	1.44E+03	4.28E+02	1.68E+02	9.02E+01	**1.93E+01**
	Std.	3.84E+02	1.96E+04	2.22E+04	3.36E+03	3.78E+01	2.96E+01	6.69E+02	1.77E+02	1.91E+02	8.20E+01	**3.79E+00**
	Best	2.24E+02	2.77E+04	1.51E+04	2.49E+03	**1.08E+01**	1.91E+01	3.41E+02	2.03E+02	2.75E+01	3.06E+01	1.40E+01
	Rank	7	11	10	9	2	3	8	6	5	4	1
F21	Mean	3.69E+05	1.81E+06	1.28E+06	3.39E+05	3.67E+03	**2.50E+03**	1.19E+05	2.20E+04	6.20E+03	9.87E+03	2.61E+03
	Std.	1.57E+05	9.57E+05	6.62E+05	1.22E+05	3.16E+03	**2.80E+03**	1.00E+05	2.22E+04	3.02E+03	1.32E+04	3.87E+03
	Best	1.26E+05	1.99E+05	1.98E+05	1.25E+05	6.53E+02	**2.21E+02**	3.16E+04	4.10E+03	2.22E+03	1.22E+03	4.39E+02
	Rank	9	11	10	8	3	1	7	6	4	5	2
F22	Mean	5.75E+02	6.84E+02	6.77E+02	2.70E+02	2.96E+02	1.36E+02	3.27E+02	3.14E+02	1.79E+02	1.14E+02	**9.70E+01**
	Std.	3.23E+02	2.74E+02	1.40E+02	**5.98E+01**	8.12E+01	7.92E+01	1.33E+02	1.41E+02	8.11E+01	7.56E+01	6.36E+01
	Best	2.71E+02	3.93E+02	5.32E+02	1.59E+02	9.35E+01	2.51E+01	5.82E+01	1.49E+02	5.06E+01	**2.41E+01**	2.67E+01
	Rank	9	11	10	5	6	3	8	7	4	2	1
F23	Mean	3.25E+02	3.78E+02	3.26E+02	3.15E+02	**3.14E+02**	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02
	Std.	5.96E+00	4.18E+01	2.45E+00	2.29E-01	2.89E-13	**0.00E+00**	1.78E-10	4.43E-01	2.92E-07	1.44E-13	2.84E-14
	Best	3.15E+02	3.26E+02	3.23E+02	3.15E+02	**3.14E+02**	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02
	Rank	9	11	10	2	1	2	2	2	2	2	2
F24	Mean	2.55E+02	2.59E+02	2.48E+02	2.27E+02	2.27E+02	2.25E+02	**2.00E+02**	**2.00E+02**	2.27E+02	2.26E+02	2.24E+02
	Std.	4.90E+00	8.93E+00	1.14E+01	1.01E+00	4.59E+00	5.21E-01	2.19E-03	**9.68E-04**	2.42E+00	1.28E+00	6.30E-01
	Best	2.48E+02	2.50E+02	2.30E+02	2.25E+02	2.24E+02	2.24E+02	**2.00E+02**	**2.00E+02**	2.25E+02	2.24E+02	2.23E+02
	Rank	10	11	9	6	6	4	1	1	6	5	3
F25	Mean	2.03E+02	2.16E+02	2.14E+02	2.10E+02	**2.00E+02**	2.06E+02	**2.00E+02**	2.02E+02	2.07E+02	2.09E+02	2.04E+02
	Std.	1.55E+00	2.63E+00	3.52E+00	1.62E+00	1.46E-01	2.49E+00	**5.33E-06**	3.62E+00	6.48E-01	3.10E+00	1.01E+00
	Best	2.02E+02	2.13E+02	2.09E+02	2.08E+02	**2.00E+02**	2.03E+02	**2.00E+02**	**2.00E+02**	2.06E+02	2.04E+02	2.03E+02
	Rank	4	11	10	9	1	6	1	3	7	8	5
F26	Mean	1.41E+02	1.21E+02	1.21E+02	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	1.10E+02	**1.00E+02**	**1.00E+02**	**1.00E+02**
	Std.	5.51E+01	4.54E+01	4.52E+01	**3.00E-02**	6.12E-02	7.09E-02	1.57E-01	3.15E+01	5.83E-02	7.52E-02	4.27E-02
	Best	**1.00E+02**	1.01E+02	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**
	Rank	11	9	9	1	1	1	1	8	1	1	1
F27	Mean	7.26E+02	6.01E+02	6.07E+02	4.38E+02	5.65E+02	**3.62E+02**	6.56E+02	7.94E+02	4.09E+02	4.34E+02	3.81E+02
	Std.	1.98E+02	2.41E+02	1.71E+02	1.33E+01	1.73E+02	5.04E+01	2.13E+02	2.15E+02	**3.71E+00**	6.71E+01	4.53E+01
	Best	4.08E+02	4.16E+02	4.15E+02	4.17E+02	4.01E+02	**3.00E+02**	4.02E+02	4.06E+02	4.04E+02	3.71E+02	**3.00E+02**
	Rank	10	7	8	5	6	1	9	11	3	4	2
F28	Mean	5.07E+02	1.44E+03	9.96E+02	9.21E+02	**3.83E+02**	8.51E+02	1.19E+03	1.43E+03	8.77E+02	8.34E+02	8.51E+02
	Std.	9.67E+01	3.31E+02	8.72E+01	3.47E+01	**5.59E+00**	2.76E+01	2.30E+02	4.37E+02	1.66E+01	3.87E+01	4.23E+01
	Best	4.09E+02	9.95E+02	9.10E+02	8.64E+02	**3.76E+02**	8.06E+02	8.94E+02	9.96E+02	8.54E+02	7.87E+02	7.91E+02
	Rank	2	11	8	7	1	4	9	10	6	3	4
F29	Mean	2.17E+02	2.73E+05	4.22E+04	3.11E+04	**2.15E+02**	9.67E+02	3.93E+06	3.08E+06	1.41E+03	1.30E+03	9.88E+02
	Std.	5.73E+00	2.09E+05	7.21E+04	1.52E+04	**1.39E+00**	1.47E+02	5.14E+06	4.99E+06	1.89E+02	5.91E+02	1.75E+02
	Best	**2.12E+02**	2.04E+04	2.46E+03	7.11E+03	2.13E+02	6.69E+02	1.44E+03	9.91E+02	1.21E+03	7.75E+02	7.83E+02
	Rank	2	9	8	7	1	3	11	10	6	5	4
F30	Mean	6.80E+02	6.07E+04	1.48E+04	7.94E+03	**3.87E+02**	1.16E+03	3.38E+03	6.48E+03	2.56E+03	1.93E+03	1.26E+03
	Std.	2.20E+02	5.73E+04	6.40E+03	2.72E+03	**6.82E+01**	5.64E+02	1.76E+03	3.43E+03	7.49E+02	5.26E+02	6.37E+02
	Best	4.95E+02	3.59E+03	8.43E+03	5.62E+03	**2.92E+02**	5.90E+02	1.59E+03	3.38E+03	1.42E+03	1.16E+03	6.22E+02
	Rank	2	11	10	9	1	3	7	8	6	5	4
Average Rank 7.90	9.80	8.57	6.40	3.23	3.80	6.70	7.50	4.57	3.40	2.23		

5.4. Comparison of the convergence performance

This section discusses the convergence characteristics of eleven algorithms on several 50D benchmark functions. Fig. 3 contains thirteen representative convergence graphs for functions F1, F2, and F3 (unimodal functions), F4, F13, and F15 (simple multimodal functions), F17, F18, F19, F20, F21, and F22 (hybrid functions), and F26 (composition function).

For F1, F2, F3, F4, F13, F15, F17, F19, F20, F21, and F22, ADELI converges faster than all other algorithms, and significantly improves the results on unimodal functions (F1, F2 and F3). For F18 and F26, the convergence curves of ADELI rapidly drop off at a relatively small position in the early stage of the optimization, and later it converges slowly, but eventually ADELI performs the best. As indicated in Fig. 3, ADELI achieves the fastest convergence speed and presents the best local search capability, which is superior to the other algorithms for these problems.

5.5. Comparison of statistical tests

To make a thorough comparison, the *t* test, *F* test, and Duncan's post hoc test [2] were used to statistically analyze the difference between ADELI and the other algorithms.

Table 5
Comparative results of eleven algorithms on 50D test functions (the boldface is the best solution).

Func.	Al.	ACOR	QEA	aQIGA	CLPSO	jDE	SaDE	ETLBO	DGSTLBO	BSA	LBSA	ADELI
F1	Mean	4.68E+07	1.25E+08	4.12E+07	4.47E+07	1.53E+06	1.75E+06	2.15E+06	2.24E+06	4.68E+06	7.28E+05	**6.13E+04**
	Std.	2.75E+07	4.18E+07	1.45E+07	1.07E+07	4.19E+05	3.58E+05	1.03E+06	8.88E+05	1.34E+06	1.97E+05	**2.29E+04**
	Best	1.10E+07	6.53E+07	2.93E+07	3.73E+07	1.01E+06	1.14E+06	8.98E+05	9.37E+05	2.73E+06	4.58E+05	**3.77E+04**
	Rank	10	11	8	9	3	4	5	6	7	2	1
F2	Mean	1.71E+09	9.78E+09	2.43E+09	7.52E+04	1.08E+01	3.84E+03	4.27E+03	8.88E+03	4.11E+03	1.06E-03	**1.17E-14**
	Std.	1.63E+09	3.46E+09	7.00E+08	1.38E+04	9.38E+00	3.46E+03	4.40E+03	1.03E+04	4.02E+03	1.08E-03	**1.72E-14**
	Best	3.69E+08	5.49E+09	1.37E+09	5.40E+04	9.35E-01	1.07E+02	2.64E+02	1.23E+01	3.09E+02	3.74E-04	**1.95E-16**
	Rank	9	11	10	8	3	4	6	7	5	2	1
F3	Mean	9.66E+03	6.16E+04	2.23E+04	1.10E+04	8.99E+00	4.29E+03	5.13E+03	3.67E+04	7.23E+02	5.95E-01	**5.54E-05**
	Std.	3.29E+03	2.17E+04	7.56E+03	3.73E+03	7.02E+00	2.28E+03	2.79E+03	4.02E+03	3.92E+02	7.58E-01	**1.02E-04**
	Best	5.99E+03	3.31E+04	1.38E+04	6.90E+03	2.76E+00	1.34E+03	1.40E+03	3.24E+04	2.54E+02	4.59E-03	**4.73E-07**
	Rank	7	11	9	8	3	5	6	10	4	2	1
F4	Mean	6.00E+02	9.74E+02	5.68E+02	1.48E+02	9.22E+01	1.11E+02	1.77E+02	9.18E+01	1.23E+02	7.51E+01	**1.96E+01**
	Std.	2.36E+02	3.19E+02	9.69E+01	1.69E+01	2.74E+00	3.36E+01	2.98E+01	4.14E+01	2.24E+01	3.46E+00	3.92E+01
	Best	3.76E+02	6.75E+02	4.83E+02	1.26E+02	8.82E+01	7.53E+01	1.37E+02	3.34E+01	9.22E+01	7.05E+01	**2.62E-11**
	Rank	10	11	9	7	4	5	8	3	6	2	1
F5	Mean	2.12E+01	2.02E+01	2.02E+01	2.08E+01	2.08E+01	2.08E+01	**2.00E+01**	2.11E+01	2.05E+01	2.04E+01	2.01E+01
	Std.	3.73E-02	2.82E-02	6.83E-02	6.20E-02	2.04E-02	6.48E-02	4.73E-02	4.66E-02	**1.55E-02**	4.54E-02	3.33E-02
	Best	2.11E+01	2.01E+01	2.01E+01	2.07E+01	2.08E+01	2.07E+01	**2.00E+01**	2.11E+01	2.04E+01	2.03E+01	**2.00E+01**
	Rank	11	3	3	7	7	7	1	10	6	5	2
F6	Mean	3.38E+01	4.22E+01	3.53E+01	4.27E+01	**2.28E+00**	1.91E+01	3.44E+01	3.38E+01	3.36E+01	2.16E+01	2.48E+00
	Std.	3.94E+00	2.46E+00	4.10E+00	1.71E+00	2.78E+00	4.25E+00	3.16E+00	5.33E+00	**1.01E+00**	5.12E+00	2.04E+00
	Best	2.98E+01	4.01E+01	3.09E+01	4.01E+01	**5.09E-04**	1.47E+01	2.93E+01	2.71E+01	3.21E+01	1.62E+01	1.45E-02
	Rank	6	10	9	11	1	3	8	6	5	4	2
F7	Mean	1.33E+01	8.96E+01	5.20E+01	5.53E-02	**1.33E-16**	1.91E-02	9.65E-02	9.34E-03	1.09E-02	4.92E-03	2.25E-02
	Std.	8.47E+00	4.91E+01	1.63E+01	1.93E-02	**4.44E-17**	2.61E-02	1.06E-01	1.43E-02	3.47E-03	6.79E-03	3.70E-02
	Best	5.67E+00	4.74E+01	3.92E+01	3.77E-02	1.11E-16	2.46E-11	9.97E-03	5.69E-09	7.53E-03	8.88E-16	**0.00E+00**
	Rank	9	11	10	7	1	5	8	3	4	2	6
F8	Mean	2.91E+02	1.27E+02	8.95E+01	1.23E+00	3.31E+01	5.97E-01	1.02E+01	2.59E+02	4.05E+00	**7.79E-11**	1.55E+01
	Std.	6.28E+01	1.97E+01	1.17E+01	6.65E-01	2.72E+00	7.96E-01	4.37E+00	1.18E+02	1.78E+00	**8.45E-11**	6.59E+00
	Best	2.11E+02	9.61E+01	8.04E+01	5.15E-01	2.90E+01	**0.00E+00**	4.98E+00	1.73E+02	1.43E+00	1.30E-12	4.97E+00
	Rank	11	9	8	3	7	2	5	10	4	1	6
F9	Mean	4.02E+02	3.24E+02	2.03E+02	2.20E+02	2.40E+02	**8.66E+01**	1.77E+02	4.39E+02	1.35E+02	1.08E+02	1.54E+02
	Std.	8.47E+01	4.64E+01	3.56E+01	7.71E+00	1.83E+01	1.48E+01	2.16E+01	1.03E+02	1.19E+01	2.08E+01	6.20E+01
	Best	2.68E+02	2.76E+02	1.52E+02	2.10E+02	2.07E+02	6.37E+01	1.45E+02	2.37E+02	1.22E+02	8.52E+01	**6.07E+01**
	Rank	10	9	6	7	8	1	5	11	3	2	4
F10	Mean	4.31E+03	3.32E+03	1.64E+03	1.50E+02	1.22E+03	**6.51E+00**	2.49E+02	1.20E+04	5.25E+01	1.52E+01	5.04E+02
	Std.	4.46E+02	4.90E+02	5.92E+02	2.40E+01	2.94E+02	6.23E+00	1.42E+02	3.01E+02	1.10E+01	**6.06E+00**	3.32E+02
	Best	3.83E+03	3.00E+03	1.03E+03	1.23E+02	9.02E+02	**9.42E-01**	8.20E+00	1.17E+04	3.29E+01	3.77E+00	2.22E+01
	Rank	10	9	8	4	7	1	5	11	3	2	6
F11	Mean	8.80E+03	6.31E+03	5.98E+03	8.38E+03	1.00E+04	9.11E+03	5.67E+03	1.34E+04	5.82E+03	**4.88E+03**	6.41E+03
	Std.	4.42E+03	5.06E+02	5.42E+02	4.49E+02	**1.36E+02**	2.33E+02	4.84E+02	3.32E+02	3.87E+02	3.79E+02	3.31E+02
	Best	4.39E+03	5.80E+03	5.20E+03	7.77E+03	9.81E+03	8.77E+03	4.96E+03	1.28E+04	5.13E+03	**4.16E+03**	6.10E+03
	Rank	8	5	4	7	10	9	2	11	3	1	6
F12	Mean	3.62E+00	4.43E-01	2.48E-01	1.10E+00	1.41E+00	1.25E+00	**1.56E-01**	3.58E+00	5.32E-01	4.04E-01	2.89E-01
	Std.	2.10E-01	6.24E-02	8.26E-02	1.15E-01	1.69E-01	1.92E-01	6.16E-02	3.14E-01	**3.84E-02**	5.11E-02	1.38E-01
	Best	3.38E+00	3.79E-01	1.59E-01	9.31E-01	1.20E+00	8.73E-01	**8.20E-02**	3.10E+00	4.57E-01	3.37E-01	1.26E-01
	Rank	11	5	2	7	9	8	1	10	6	4	3
F13	Mean	5.77E-01	8.00E-01	7.17E-01	4.89E-01	4.17E-01	5.22E-01	5.66E-01	7.34E-01	4.22E-01	4.35E-01	**2.79E-01**
	Std.	1.16E-01	1.12E-01	1.12E-01	**2.60E-02**	4.98E-02	4.11E-02	6.51E-02	8.95E-02	4.59E-02	5.09E-02	3.48E-02
	Best	4.46E-01	6.93E-01	6.02E-01	4.46E-01	3.32E-01	4.56E-01	4.68E-01	6.24E-01	3.59E-01	3.81E-01	**2.44E-01**
	Rank	8	11	9	5	2	6	7	10	3	4	1
F14	Mean	5.32E+00	1.60E+01	5.25E+00	3.73E-01	3.10E-01	3.08E-01	3.07E-01	3.24E-01	3.17E-01	**2.99E-01**	3.57E-01
	Std.	6.82E+00	1.12E+01	6.36E+00	2.88E-02	**1.46E-02**	3.33E-02	3.34E-02	4.48E-02	2.22E-02	3.80E-02	1.33E-01
	Best	3.60E-01	1.03E+00	3.39E-01	3.38E-01	2.92E-01	2.74E-01	2.59E-01	2.56E-01	2.96E-01	2.50E-01	**2.27E-01**
	Rank	10	11	9	8	4	3	2	6	5	1	7
F15	Mean	3.30E+03	8.84E+03	6.38E+02	2.66E+01	2.31E+01	2.40E+01	5.60E+01	4.90E+01	1.80E+01	2.37E+01	**1.27E+01**
	Std.	5.02E+03	4.21E+03	5.62E+02	**1.52E+00**	1.83E+00	4.97E+00	1.89E+01	5.78E+00	2.43E+00	3.93E+00	5.67E+00
	Best	2.10E+02	1.69E+03	1.26E+02	2.49E+01	2.06E+01	1.65E+01	3.29E+01	4.24E+01	1.53E+01	1.77E+01	**6.99E+00**
	Rank	10	11	9	6	3	5	8	7	2	4	1
F16	Mean	2.16E+01	2.00E+01	**1.84E+01**	2.13E+01	2.08E+01	2.06E+01	1.86E+01	2.19E+01	1.90E+01	1.87E+01	2.04E+01
	Std.	8.28E-01	9.60E-01	3.98E-01	**1.95E-01**	2.25E-01	2.25E-01	4.01E-01	3.79E-01	4.28E-01	3.48E-01	3.67E-01
	Best	2.02E+01	1.89E+01	**1.79E+01**	2.09E+01	2.06E+01	2.02E+01	1.82E+01	2.14E+01	1.84E+01	1.82E+01	1.98E+01
	Rank	10	5	1	9	8	7	2	11	4	3	6
F17	Mean	9.36E+06	2.54E+07	1.37E+07	5.00E+06	1.11E+05	7.56E+04	1.54E+05	1.36E+06	4.03E+05	6.06E+04	**2.17E+04**
	Std.	3.59E+06	1.13E+07	7.45E+06	8.95E+05	6.41E+04	4.89E+04	5.74E+04	5.16E+05	1.10E+05	2.73E+04	**1.63E+04**
	Best	3.16E+06	1.69E+07	3.22E+06	3.73E+06	5.12E+04	3.79E+04	1.03E+05	4.69E+05	2.44E+05	2.62E+04	**8.79E+03**
	Rank	9	11	10	8	4	3	5	7	6	2	1
F18	Mean	1.44E+04	6.61E+08	6.08E+07	3.49E+03	5.11E+02	5.98E+02	2.18E+03	1.76E+03	7.54E+02	1.48E+03	**2.19E+02**
	Std.	1.94E+04	5.81E+08	3.17E+07	**2.08E+02**	4.03E+02	5.57E+02	1.44E+03	1.47E+03	5.72E+02	1.07E+03	2.66E+02

(*continued on next page*)

Table 5 (*continued*)

	Best	1.84E+03	1.12E+08	1.19E+07	3.18E+03	9.77E+01	1.12E+02	2.78E+02	3.56E+02	1.49E+02	3.57E+02	**3.28E+01**
	Rank	9	11	10	8	2	3	7	6	4	5	1
F19	Mean	5.34E+01	1.68E+02	8.88E+01	2.44E+01	1.75E+01	2.75E+01	1.94E+01	2.83E+01	4.54E+01	1.66E+01	**9.53E+00**
	Std.	2.84E+01	8.30E+01	2.30E+01	1.48E+00	8.89E+00	1.16E+01	1.84E+00	1.08E+01	1.09E+01	2.13E+00	1.75E+00
	Best	2.42E+01	1.01E+02	4.86E+01	2.31E+01	1.19E+01	1.27E+01	1.70E+01	1.67E+01	3.62E+01	1.44E+01	**6.95E+00**
	Rank	9	11	10	5	3	6	4	7	8	2	1
F20	Mean	2.27E+03	7.64E+04	3.22E+04	1.06E+04	3.10E+02	2.91E+03	4.84E+03	6.44E+03	2.42E+03	3.13E+02	**1.67E+02**
	Std.	1.52E+03	3.95E+04	1.30E+04	4.51E+03	**4.48E+01**	2.19E+03	2.20E+03	3.33E+03	1.01E+03	1.76E+02	9.02E+01
	Best	6.90E+02	5.00E+04	1.92E+04	2.90E+03	2.48E+02	1.10E+03	2.50E+03	2.49E+03	8.53E+02	1.36E+02	**5.78E+01**
	Rank	4	11	10	9	2	6	7	8	5	3	1
F21	Mean	8.54E+06	1.78E+07	1.17E+07	2.00E+06	5.34E+04	5.44E+04	8.72E+04	5.35E+05	1.73E+05	5.20E+04	**1.04E+04**
	Std.	9.23E+06	7.22E+06	5.59E+06	3.79E+05	3.07E+04	1.86E+04	3.79E+04	1.47E+05	1.18E+05	3.64E+04	**4.84E+03**
	Best	3.71E+05	9.75E+06	6.43E+06	1.57E+06	1.49E+04	2.85E+04	4.68E+04	3.10E+05	6.59E+04	1.53E+04	**3.01E+03**
	Rank	9	11	10	8	3	4	5	7	6	2	1
F22	Mean	1.10E+03	1.50E+03	1.09E+03	9.51E+02	5.62E+02	4.68E+02	1.31E+03	1.61E+03	7.92E+02	6.92E+02	**3.85E+02**
	Std.	2.22E+02	2.01E+02	3.16E+02	7.93E+01	2.37E+02	7.62E+01	3.42E+02	1.66E+02	**6.20E+01**	1.63E+02	1.77E+02
	Best	8.57E+02	1.27E+03	7.14E+02	8.72E+02	3.64E+02	3.90E+02	9.32E+02	1.39E+03	6.90E+02	4.50E+02	**1.43E+02**
	Rank	8	10	7	6	3	2	9	11	5	4	1
F23	Mean	3.49E+02	5.03E+02	3.85E+02	3.44E+02	3.44E+02	3.44E+02	**2.00E+02**	**2.00E+02**	3.44E+02	3.44E+02	3.44E+02
	Std.	1.01E+01	5.52E+01	2.96E+01	3.72E-02	2.54E-14	4.40E-14	2.46E-13	**0.00E+00**	8.57E-08	5.08E-14	2.54E-14
	Best	3.40E+02	4.19E+02	3.48E+02	3.44E+02	3.44E+02	3.44E+02	**2.00E+02**	**2.00E+02**	3.44E+02	3.44E+02	3.44E+02
	Rank	9	11	10	3	3	3	1	1	3	3	3
F24	Mean	2.96E+02	3.12E+02	2.99E+02	2.65E+02	2.68E+02	2.76E+02	**2.00E+02**	**2.00E+02**	2.67E+02	2.75E+02	2.76E+02
	Std.	1.83E+01	6.24E+00	1.19E+01	1.71E-01	1.67E+00	3.99E+00	1.20E-01	**2.69E-05**	2.45E+00	8.47E+00	1.63E+00
	Best	2.76E+02	3.07E+02	2.79E+02	2.65E+02	2.65E+02	2.70E+02	**2.00E+02**	**2.00E+02**	2.64E+02	2.59E+02	2.75E+02
	Rank	9	11	10	3	5	7	1	1	4	6	7
F25	Mean	2.13E+02	2.37E+02	2.28E+02	2.21E+02	2.09E+02	2.15E+02	**2.00E+02**	**2.00E+02**	2.14E+02	2.24E+02	2.08E+02
	Std.	8.01E+00	7.89E+00	4.95E+00	1.33E+00	1.11E+00	1.20E+01	4.12E-13	**0.00E+00**	6.44E+00	8.14E+00	1.26E+00
	Best	2.05E+02	2.28E+02	2.22E+02	2.19E+02	2.08E+02	**2.00E+02**	**2.00E+02**	**2.00E+02**	2.02E+02	2.14E+02	2.06E+02
	Rank	5	11	10	8	4	7	1	1	6	9	3
F26	Mean	2.19E+02	1.71E+02	2.04E+02	**1.00E+02**	1.60E+02	1.40E+02	1.80E+02	2.00E+02	**1.00E+02**	1.40E+02	**1.00E+02**
	Std.	1.62E+02	5.65E+01	1.96E+00	1.05E-02	4.89E+01	4.88E+01	3.98E+01	**0.00E+00**	4.86E-02	4.89E+01	4.18E-02
	Best	**1.00E+02**	1.04E+02	2.03E+02	**1.00E+02**	**1.00E+02**	**1.00E+02**	1.01E+02	2.00E+02	**1.00E+02**	**1.00E+02**	**1.00E+02**
	Rank	11	7	10	1	6	4	8	9	1	4	1
F27	Mean	1.28E+03	1.54E+03	1.31E+03	1.04E+03	3.98E+02	8.22E+02	**2.00E+02**	3.80E+02	1.13E+03	8.73E+02	3.77E+02
	Std.	9.99E+01	1.37E+02	7.56E+01	2.85E+02	4.51E+01	3.39E+01	**3.50E-13**	3.61E+02	6.72E+01	7.08E+01	4.83E+01
	Best	1.10E+03	1.40E+03	1.24E+03	7.20E+02	3.33E+02	7.72E+02	**2.00E+02**	**2.00E+02**	1.07E+03	7.47E+02	3.00E+02
	Rank	9	11	10	7	4	5	1	3	8	6	2
F28	Mean	5.51E+02	4.63E+03	2.50E+03	1.33E+03	1.28E+03	1.48E+03	**2.00E+02**	**2.00E+02**	1.33E+03	1.34E+03	1.20E+03
	Std.	1.55E+02	7.81E+02	6.40E+02	5.12E+01	4.21E+01	5.60E+01	4.33E-13	**0.00E+00**	3.34E+01	3.30E+01	3.75E+01
	Best	4.11E+02	3.80E+03	1.96E+03	1.26E+03	1.20E+03	1.41E+03	**2.00E+02**	**2.00E+02**	1.29E+03	1.29E+03	1.13E+03
	Rank	3	11	10	6	5	9	1	1	6	8	4
F29	Mean	**2.76E+02**	8.77E+06	1.39E+06	1.26E+06	1.73E+03	1.43E+03	1.41E+07	3.28E+07	2.32E+03	1.07E+03	1.74E+03
	Std.	**3.54E+01**	6.95E+06	4.93E+05	2.09E+06	2.92E+02	2.75E+02	1.73E+07	4.18E+07	4.24E+02	2.71E+02	4.45E+01
	Best	2.44E+02	1.82E+06	6.26E+05	3.43E+04	1.48E+03	1.18E+03	**2.00E+02**	**2.00E+02**	1.74E+03	7.51E+02	1.69E+03
	Rank	1	9	8	7	4	3	10	11	6	2	5
F30	Mean	**1.69E+03**	1.19E+05	4.72E+04	1.19E+04	9.40E+03	1.19E+04	1.31E+04	8.34E+03	1.04E+04	1.01E+04	9.28E+03
	Std.	4.16E+02	3.87E+04	2.58E+04	1.20E+03	**3.16E+02**	1.41E+03	2.36E+03	6.77E+03	2.10E+03	6.54E+02	3.25E+02
	Best	1.36E+03	6.65E+04	2.08E+04	1.02E+04	8.98E+03	9.98E+03	9.61E+03	**2.00E+02**	9.00E+03	9.61E+03	8.82E+03
	Rank	1	11	10	7	4	7	9	2	6	5	3
Average Rank 8.20	9.67	8.30	6.63	4.40	4.80	4.93	6.90	4.80	3.40	2.93		

5.5.1. t *test comparisons for 30D and 50D functions*

The *t* test adopts a two-tailed test with a significance level of 0.05. Table 6 shows the *t* values and the *P* values on 30D functions between the ADELI and other algorithms. The better solutions between ADELI and other algorithms on thirty functions are shown in boldface. The symbols '+', '−', and '=' denote that ADELI performed significantly better than, significantly worse than, and almost the same as the algorithm under comparison, respectively. The average excellent and good rate, defined by $\sum_{i=1}^{12}((+_i)+(=_i))/(30\times 12)$ [7], between ADELI and the other algorithms on 30D functions is 95.56%. Table 7 presents *t* test results on 50D functions between the ADELI and other algorithms. The average excellent and good rate between ADELI and other algorithms on 50D functions is 90.28%. Hence, ADELI performs the best on both 30D and 50D functions, especially on 30D functions.

5.5.1. F *test comparisons for 30D and 50D functions*

In *F* test experiments, the hypothesis assumptions can be expressed as follows. H0: there is no difference among the algorithms; H1: there is at least one algorithm that has a different result from the other algorithms. There are eleven algorithms for thirty functions, which means the degree of freedom is ten, and the sample size is thirty. If we choose the significance level of 0.05, then the critical value is 2.4663, which can be found using the F-distribution table.

Table 6
Comparison between ADELI and other algorithms using a t -test for 30D functions in the CEC 2014 sets.

Func.		ACOR	QEA	aQIGA	CLPSO	jDE	SaDE	ETLBO	DGSTLBO	BSA	LBSA
F1	T	**5.20153**	**2.33505**	**4.05826**	**13.68748**	**3.57523**	**2.98687**	**3.80204**	**2.68804**	**3.68522**	**2.10622**
	P	**0.00082**	**0.04778**	**0.00364**	**0.00000**	**0.00724**	**0.01742**	**0.00522**	**0.02758**	**0.00617**	**0.06827**
F2	T	1.17749	**9.37489**	1.57807	**5.41812**	**2.59869**	**3.66676**	**3.43888**	0.62594	0.42011	**4.46045**
	P	0.27284	**0.00001**	0.15320	**0.00063**	**0.03168**	**0.00634**	**0.00884**	0.54879	0.68546	**0.00211**
F3	T	1.30242	**2.36418**	**2.67001**	**3.89806**	**3.58671**	**2.16999**	**7.17561**	**2.25831**	0.63978	**3.94526**
	P	0.22901	**0.04566**	**0.02836**	**0.00456**	**0.00712**	**0.06182**	**0.00009**	**0.05386**	0.54020	**0.00426**
F4	T	**8.90727**	**5.76187**	**11.32706**	**15.80558**	**3.56610**	**16.29924**	**39.95512**	**11.01581**	**16.88283**	**2.07709**
	P	**0.00002**	**0.00042**	**0.00000**	**0.00000**	**0.00734**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.07144**
F5	T	**23.74305**	**4.29919**	**3.57030**	**37.40487**	**28.60539**	**18.57449**	**54.36840**	**62.05680**	**157.22003**	**22.15365**
	P	**0.00000**	**0.00262**	**0.00729**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**
F6	T	**14.05467**	**25.25992**	**13.33694**	**23.56942**	**6.15482**	-1.25666	**10.39613**	**10.67574**	**20.78028**	**4.31945**
	P	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00027**	0.24433	**0.00001**	**0.00001**	**0.00000**	**0.00255**
F7	T	**4.23931**	**6.42224**	**5.27185**	-0.61155	-1.23835	-1.23835	1.16012	**2.17125**	0.09258	-0.37876
	P	**0.00284**	**0.00020**	**0.00075**	0.55781	0.25069	0.25069	0.27945	**0.06170**	0.92851	0.71472
F8	T	**14.80081**	**27.50528**	**8.20288**	**2.48408**	-0.47139	**2.26185**	**42.64454**	**4.41167**	**4.64314**	**65.00000**
	P	**0.00000**	**0.00000**	**0.00004**	**0.03787**	0.64995	**0.05357**	**0.00000**	**0.00225**	**0.00166**	**0.00000**
F9	T	**5.67557**	**4.78428**	**2.15844**	0.54509	-1.11948	0.62937	**2.10995**	**2.55060**	-0.91422	-2.24085
	P	**0.00047**	**0.00138**	**0.06294**	0.60056	0.29542	0.54666	**0.06788**	**0.03414**	0.38733	0.05535
F10	T	**7.57158**	**12.46825**	**7.19152**	**5.91946**	-4.60613	-3.05645	**10.49183**	**16.80813**	**3.88749**	0.50349
	P	**0.00006**	**0.00000**	**0.00009**	**0.00035**	0.00174	0.01567	**0.00001**	**0.00000**	**0.00463**	0.62819
F11	T	-0.13882	1.20882	**2.26810**	1.71067	0.15818	**3.01215**	**12.05543**	0.78376	-0.42558	-1.76727
	P	0.89302	0.26125	**0.05305**	0.12551	0.87824	**0.01676**	**0.00000**	0.45576	0.68163	0.11516
F12	T	**21.49957**	0.56772	0.45419	**3.78163**	**3.72206**	**9.54284**	**13.80324**	**25.03244**	1.66172	1.46226
	P	**0.00000**	0.58580	0.66176	**0.00538**	**0.00585**	**0.00001**	**0.00000**	**0.00000**	0.13514	0.18181
F13	T	**4.81681**	**6.25451**	**8.61800**	**9.69026**	**2.45120**	**2.60284**	**7.22314**	**6.88327**	1.48681	1.58516
	P	**0.00133**	**0.00024**	**0.00003**	**0.00001**	**0.03986**	**0.03148**	**0.00009**	**0.00013**	0.17538	0.15159
F14	T	1.90500	**2.59380**	**3.16277**	**5.31162**	1.14226	**2.04792**	1.14596	**3.56715**	-0.14731	-0.48725
	P	0.09325	**0.03192**	**0.01334**	**0.00072**	0.28638	**0.07475**	0.28493	**0.00732**	0.88653	0.63915
F15	T	1.87487	1.76844	1.33009	**7.78382**	**2.29144**	**5.78471**	**2.64885**	**3.61155**	**3.30804**	**2.35283**
	P	0.09767	0.11496	0.22016	**0.00005**	**0.05115**	**0.00041**	**0.02931**	**0.00687**	**0.01073**	**0.04648**
F16	T	**2.11396**	**3.17760**	-0.69514	**2.58290**	-0.00794	**3.75359**	**3.74462**	**2.06676**	0.76989	-0.46534
	P	**0.06746**	**0.01304**	0.50665	**0.03247**	0.99386	**0.00560**	**0.00567**	**0.07259**	0.46349	0.65409
F17	T	**3.94229**	**2.53881**	**4.74255**	**15.50377**	**3.23042**	1.03468	**4.48586**	0.53620	**2.22208**	**4.01014**
	P	**0.00428**	**0.03478**	**0.00146**	**0.00000**	**0.01205**	0.33108	**0.00204**	0.60640	**0.05700**	**0.00389**
F18	T	**2.08239**	**3.60027**	1.64084	**13.88654**	**3.09340**	**5.12897**	**4.15407**	1.91486	1.19406	**3.34612**
	P	**0.07085**	**0.00698**	0.13946	**0.00000**	**0.01481**	**0.00090**	**0.00319**	0.09184	0.26666	**0.01014**
F19	T	1.27236	**3.78405**	**3.92972**	**15.50576**	**20.62731**	**4.32654**	0.94813	**2.15734**	**9.55846**	**7.27053**
	P	0.23898	**0.00536**	**0.00436**	**0.00000**	**0.00000**	**0.00252**	0.37082	**0.06305**	**0.00001**	**0.00009**
F20	T	**2.93735**	**5.74683**	**4.47408**	**7.91461**	1.04951	**3.22113**	**4.90780**	**4.60296**	**3.82819**	**2.72302**
	P	**0.01879**	**0.00043**	**0.00207**	**0.00005**	0.32461	**0.01222**	**0.00118**	**0.00175**	**0.00503**	**0.02613**
F21	T	**5.22077**	**4.21458**	**4.30117**	**5.24910**	-0.47222	-0.01174	**8.27364**	0.20364	1.62747	**2.34565**
	P	**0.00080**	**0.00294**	**0.00261**	**0.00077**	0.64938	0.99092	**0.00003**	0.84372	0.14229	**0.04700**
F22	T	**3.24683**	**4.67190**	**8.44834**	**4.52445**	**3.99503**	0.66965	**4.03099**	**2.30127**	1.44694	0.69160
	P	**0.01176**	**0.00160**	**0.00003**	**0.00194**	**0.00398**	0.52193	**0.00378**	**0.05037**	0.18593	0.50876
F23	T	**3.52814**	**3.35740**	**10.04146**	0.00000	-31.09000	0.00000	0.00000	-1.42101	0.00000	0.00000
	P	**0.00775**	**0.00997**	**0.00001**	0.00000	0.00000	0.00000	0.00000	0.19310	0.00000	0.00000
F24	T	**14.06281**	**8.81231**	**4.76957**	**6.18681**	0.15077	**4.10604**	-85.03299	-85.02555	**2.01851**	**4.65957**
	P	**0.00000**	**0.00002**	**0.00141**	**0.00026**	0.88389	**0.00341**	0.00000	0.00000	**0.07824**	**0.00162**
F25	T	-1.33635	**10.03883**	**6.17644**	**5.62337**	-8.10158	**3.27333**	-8.24545	-2.10252	**5.79118**	**3.52129**
	P	0.21819	**0.00001**	**0.00027**	**0.00050**	0.00004	**0.01130**	0.00004	0.06867	**0.00041**	**0.00783**
F26	T	1.64769	1.02368	1.01061	0.00000	0.00000	0.00000	0.00000	0.09053	0.00000	0.00000
	P	0.13803	0.33594	0.34179	0.00000	0.00000	0.00000	0.00000	0.93009	0.00000	0.00000
F27	T	**3.80334**	**2.00849**	**2.85571**	**3.33536**	1.43910	-0.72211	**5.00996**	**3.76537**	1.43427	0.79443
	P	**0.00521**	**0.07946**	**0.02129**	**0.01030**	0.18807	0.49079	**0.00104**	**0.00550**	0.18941	0.44987
F28	T	-7.28857	**3.95231**	**3.34137**	**2.14145**	-24.26166	0.13171	1.30946	**4.20510**	2.00194	-0.17618
	P	0.00008	**0.00422**	**0.01021**	**0.06463**	0.00000	0.89847	0.22673	**0.00298**	0.08027	0.86453
F29	T	-9.86465	**2.90170**	1.27834	**3.82494**	-9.89970	0.29686	-0.36939	**2.35948**	**2.26088**	0.55526
	P	0.00001	**0.01984**	0.23697	**0.00505**	0.00001	0.77414	0.72142	**0.04600**	**0.05365**	0.59390
F30	T	-1.89665	**2.31872**	**4.72703**	**10.03753**	-2.98967	-0.16790	**2.97683**	**10.40842**	**2.91622**	1.83598
	P	0.09445	**0.04902**	**0.00149**	**0.00001**	0.01734	0.87083	**0.01769**	**0.00001**	**0.01940**	0.10368
+		19	26	23	25	13	17	21	21	15	15
=		9	4	7	5	11	12	7	7	15	14
−		2	0	0	0	6	1	2	2	0	1

Table 7
Comparison between ADELI and other algorithms using a *t* test for 50D functions in the CEC 2014 sets.

Func.		ACOR	QEA	aQIGA	CLPSO	jDE	SaDE	ETLBO	DGSTLBO	BSA	LBSA
F1	T	**3.79763**	**6.65886**	**6.35231**	**8.35259**	**6.99670**	**9.41040**	**4.06147**	**4.89541**	**6.86526**	**6.73247**
	P	**0.00525**	**0.00016**	**0.00022**	**0.00003**	**0.00011**	**0.00001**	**0.00363**	**0.00120**	**0.00013**	**0.00015**
F2	T	**2.34951**	**6.32609**	**7.74702**	**10.90953**	**2.29815**	**2.22364**	1.94023	1.72897	**2.04189**	1.95861
	P	**0.04672**	**0.00023**	**0.00005**	**0.00000**	**0.05062**	**0.05686**	0.08831	0.12207	**0.07545**	0.08584
F3	T	**6.57677**	**6.33674**	**6.61822**	**5.87947**	**2.56245**	**3.77365**	**3.69361**	**18.25038**	**3.69135**	1.57066
	P	**0.00017**	**0.00022**	**0.00017**	**0.00037**	**0.03352**	**0.00544**	**0.00610**	**0.00000**	**0.00612**	0.15490
F4	T	**5.41049**	**6.62387**	**11.52536**	**6.00859**	**3.68945**	**3.51971**	**6.36809**	**2.53266**	**4.59831**	**2.81655**
	P	**0.00064**	**0.00017**	**0.00000**	**0.00032**	**0.00614**	**0.00785**	**0.00022**	**0.03511**	**0.00176**	**0.02261**
F5	T	**45.79390**	**3.19227**	**2.45419**	**21.48973**	**37.10955**	**20.00685**	-1.53759	**37.14951**	**20.24429**	**10.90458**
	P	**0.00000**	**0.01276**	**0.03968**	**0.00000**	**0.00000**	**0.00000**	0.16270	**0.00000**	**0.00000**	**0.00000**
F6	T	**15.41153**	**26.49922**	**15.64351**	**30.24442**	-0.11492	**7.03651**	**16.95817**	**10.95620**	**27.37690**	**6.94251**
	P	**0.00000**	**0.00000**	**0.00000**	**0.00000**	0.91134	**0.00011**	**0.00000**	**0.00000**	**0.00000**	**0.00012**
F7	T	**3.49608**	**4.07861**	**7.13673**	1.57314	-1.21639	-0.15066	1.31403	-0.66380	-0.62549	-0.93463
	P	**0.00813**	**0.00354**	**0.00010**	0.15433	0.25850	0.88397	0.22526	0.52548	0.54907	0.37733
F8	T	**9.74541**	**11.81226**	**11.97601**	-4.31775	**4.91911**	-4.49820	-1.33895	**4.12955**	-3.36327	-4.71215
	P	**0.00001**	**0.00000**	**0.00000**	0.00255	**0.00117**	0.00201	0.21738	**0.00330**	0.00988	0.00152
F9	T	**5.06095**	**4.56869**	1.41492	**2.09999**	**2.66892**	-2.11363	0.71182	**4.71827**	-0.60118	-1.39260
	P	**0.00098**	**0.00183**	0.19482	**0.06894**	**0.02841**	0.06749	0.49681	**0.00151**	0.56436	0.20122
F10	T	**14.65533**	**10.22052**	**3.63577**	-2.12588	3.24870	-2.99491	-1.41383	**51.24845**	-2.71711	-2.94239
	P	**0.00000**	**0.00001**	**0.00663**	0.06622	**0.01172**	0.01720	0.19512	**0.00000**	0.02637	0.01864
F11	T	1.20508	-0.34706	-1.50229	**7.02352**	**20.20216**	**13.32666**	-2.55983	**30.02107**	-2.33476	-6.10227
	P	0.26261	0.73750	0.17142	**0.00011**	**0.00000**	**0.00000**	0.03366	**0.00000**	0.04780	0.00029
F12	T	**28.57515**	**2.06573**	-0.52694	**8.97881**	**10.24729**	**8.12161**	-1.76337	**19.17450**	**3.39295**	1.55841
	P	**0.00000**	**0.07271**	0.61253	**0.00002**	**0.00001**	**0.00004**	0.11585	**0.00000**	**0.00946**	0.15775
F13	T	**5.44288**	**9.79812**	**8.23346**	**9.69675**	**4.56636**	**9.02040**	**7.78876**	**9.48732**	**4.98395**	**5.06465**
	P	**0.00061**	**0.00001**	**0.00004**	**0.00001**	**0.00183**	**0.00002**	**0.00005**	**0.00001**	**0.00107**	**0.00097**
F14	T	1.62706	**3.12274**	**1.72130**	**0.23760**	-0.69864	-0.72163	-0.72570	-0.46846	-0.59516	-0.84063
	P	0.14238	**0.01417**	**0.12350**	**0.81816**	0.50458	0.49107	0.48871	0.65195	0.56818	0.42497
F15	T	1.46192	**4.68149**	**2.48982**	**4.72713**	**3.48518**	**3.00956**	**4.39688**	**8.96735**	1.71665	**3.19439**
	P	0.18190	**0.00158**	**0.03753**	**0.00149**	**0.00826**	**0.01682**	**0.00230**	**0.00002**	0.12438	**0.01272**
F16	T	**2.98214**	-0.82112	-7.81609	**4.09555**	1.66039	0.92373	-6.72744	**5.82510**	-5.10150	-6.82695
	P	**0.01754**	0.43536	0.00005	**0.00346**	0.13542	0.38264	0.00015	**0.00039**	0.00093	0.00013
F17	T	**5.82090**	**5.01777**	**4.11470**	**11.12936**	**2.70649**	**2.09531**	**4.42831**	**5.19521**	**6.84872**	**2.44516**
	P	**0.00040**	**0.00103**	**0.00337**	**0.00000**	**0.02680**	**0.06944**	**0.00220**	**0.00083**	**0.00013**	**0.04024**
F18	T	1.63535	**2.54393**	**4.28982**	**19.32839**	1.20940	1.22640	**2.65402**	**2.06826**	1.69817	**2.28195**
	P	0.14061	**0.03450**	**0.00265**	**0.00000**	0.26103	0.25492	**0.02907**	**0.07242**	0.12791	**0.05191**
F19	T	**3.44416**	**4.27122**	**7.69244**	**12.94923**	1.75056	**3.06671**	**7.75841**	**3.41960**	**6.48767**	**5.11608**
	P	**0.00877**	**0.00272**	**0.00006**	**0.00000**	0.11813	**0.01542**	**0.00005**	**0.00909**	**0.00019**	**0.00091**
F20	T	**3.07223**	**4.31781**	**5.50679**	**4.65095**	**2.83465**	**2.51402**	**4.25586**	**3.76659**	**4.42590**	1.48270
	P	**0.01530**	**0.00255**	**0.00057**	**0.00164**	**0.02199**	**0.03614**	**0.00278**	**0.00549**	**0.00221**	0.17644
F21	T	**2.06708**	**5.49834**	**4.68433**	**10.51730**	**2.76633**	**4.57700**	**4.01900**	**7.14814**	**2.75746**	**2.26749**
	P	**0.07256**	**0.00057**	**0.00157**	**0.00001**	**0.02443**	**0.00181**	**0.00385**	**0.00010**	**0.02477**	**0.05310**
F22	T	**5.35857**	**8.77561**	**4.19502**	**5.83330**	1.19474	0.86118	**4.81665**	**10.12587**	**4.34256**	**2.54835**
	P	**0.00068**	**0.00002**	**0.00302**	**0.00039**	0.26641	0.41420	**0.00133**	**0.00001**	**0.00247**	**0.03426**
F23	T	1.02958	**6.45411**	**3.06892**	1.21024	0.00000	0.00000	-116.39157	-113.29510	0.00000	0.00000
	P	0.33333	**0.00020**	**0.01537**	0.26073	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
F24	T	**2.37245**	**12.29280**	**4.10225**	-13.61429	-7.17602	-0.44215	-93.32701	-93.92191	-6.29280	-0.29550
	P	**0.04507**	**0.00000**	**0.00343**	0.00000	0.00009	0.67009	0.00000	0.00000	0.00023	0.77514
F25	T	1.31960	**7.99624**	**8.68486**	**14.66149**	1.67879	1.13434	-12.52575	-12.52575	1.87450	**3.87166**
	P	0.22348	**0.00004**	**0.00002**	**0.00000**	0.13171	0.28950	0.00000	0.00000	0.09773	**0.00473**
F26	T	1.63373	**2.77591**	**118.43744**	0.00000	**2.44756**	1.63471	**4.01235**	**4764.47270**	0.00000	1.63037
	P	0.14096	**0.02408**	**0.00000**	0.00000	**0.04009**	0.14075	**0.00388**	**0.00000**	0.00000	0.14167
F27	T	**17.64046**	**17.70057**	**22.48815**	**4.59481**	0.63010	**15.06699**	-7.34577	0.01649	**18.26515**	**11.54928**
	P	**0.00000**	**0.00000**	**0.00000**	**0.00177**	0.54620	**0.00000**	0.00008	0.98725	**0.00000**	**0.00000**
F28	T	-8.97014	**9.82917**	**4.54393**	**4.15430**	**2.48162**	**8.01834**	-53.14648	-53.14648	**5.16062**	**5.92951**
	P	0.00002	**0.00001**	**0.00189**	**0.00319**	**0.03802**	**0.00004**	0.00000	0.00000	**0.00086**	**0.00035**
F29	T	-53.64528	**2.82218**	**6.28707**	1.20160	-0.05009	-2.26405	1.63281	1.56961	**2.72050**	-4.94292
	P	0.00000	**0.02242**	**0.00024**	0.26388	0.96128	0.05338	0.14115	0.15515	**0.02623**	0.00113
F30	T	-30.75237	**6.32067**	**3.29221**	**4.21406**	0.54150	**3.65531**	**3.21826**	-0.27436	1.06014	**2.31333**
	P	0.00000	**0.00023**	**0.01099**	**0.00294**	0.60291	**0.00644**	**0.01227**	0.79076	0.32003	**0.04943**
+		20	28	25	22	17	17	14	20	16	15
=		7	2	4	5	12	9	9	6	9	10
−		3	0	1	3	1	4	7	4	5	5

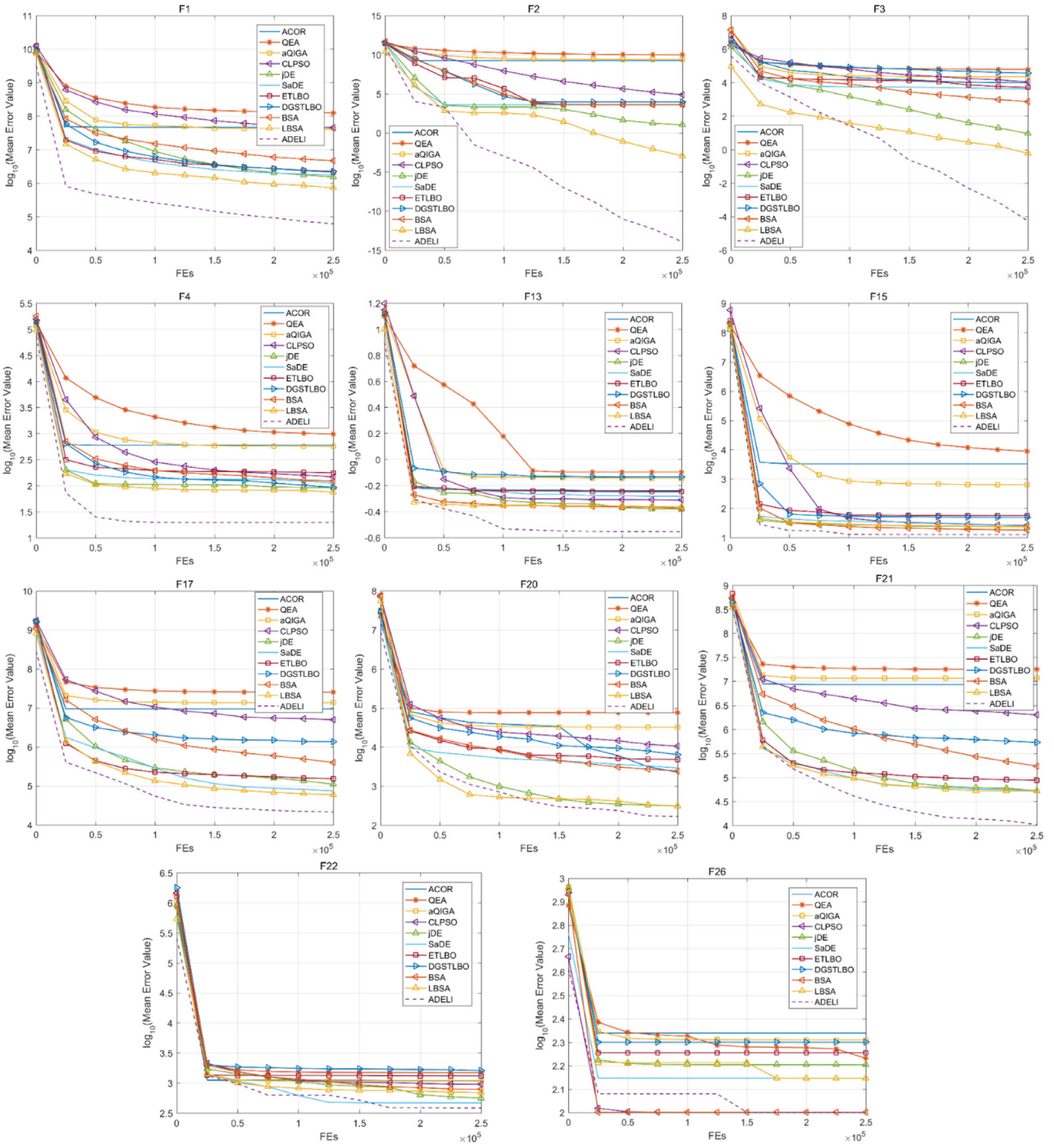

Fig. 3. The convergence curves of the eleven algorithms on thirteen 50D benchmark functions.

After the calculation, the F-scores of the mean's ranks of 30D and 50D functions are 48.255 and 35.499, respectively, which are much higher than the critical value. Hence, the hypothesis H0 is rejected as these algorithms have unequal results.

5.5.2. Duncan's post hoc test comparisons for 30D and 50D functions

As the *F* test shows that there are significant differences between these algorithms, the Duncan's post hoc test is used to justify the difference between ADELI and the other algorithms. Table 8 shows the results of the significance between ADELI and the other algorithms, which are evaluated with SPSS software. It can be concluded that, ADELI significantly outperforms the other algorithms on 30D functions, except for jDE, SaDE, and LBSA, at a significance level of 0.05. For 50D problems, ADELI is significantly superior to the other algorithms, excluding jDE, SADE, ETLBO, and LBSA.

Table 8
Comparison between ADELI and other algorithms using the Duncan's post hoc test for 30D and 50D functions in the CEC 2014 sets.

Al.	ACOR	QEA	aQIGA	CLPSO	jDE	SaDE	ETLBO	DGSTLBO	BSA	LBSA
30D	0.000	0.000	0.000	0.000	0.877	0.254	0.000	0.000	0.000	0.529
50D	0.000	0.000	0.000	0.000	0.716	0.181	0.331	0.000	0.049	1.000

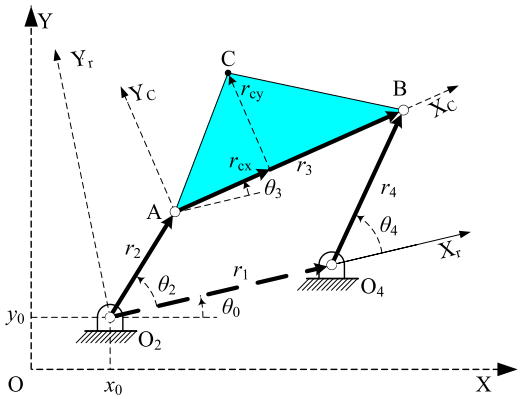

Fig. 4. Variables of the four-bar mechanism used in this experiment.

Therefore, as indicated in Tables 6–8 and as noted from the results of the *F* tests, ADELI shows a statistically significant performance compared to all other algorithms on most of the problems with 30 dimensions and 50 dimensions.

6. Application example

This section considers the optimal synthesis problem of four-bar mechanisms utilizing the five algorithms presented in Section 5.2. The four-bar mechanism is a simple yet indispensable mechanism which has been widely used in various machines and devices. The dimensional synthesis of four-bar mechanisms is aimed at synthesizing a mechanism with minimum errors between the coupler points and target points to meet the design requirements.

6.1. Problem formulation

The schematic of the four-bar mechanism together with the variables is shown in Fig. 4. It consists of four links connected in a loop by four pivots, where point O_2 and point O_4 are two fixed pivots, and point A and point B are two moving pivots. Point C is the coupler point on the coupler link. The design variable vector, *x*, is denoted by:

$$\mathrm{x} = [r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, x_0, y_0, \theta_0, \theta_2^i], i = 1...n. \tag{20}$$

where r_1, r_2, r_3 and $r_4$are the length of the ground link, input link, coupler (floating link), and output link, respectively.r_{cx} and r_{cy} are the coordinates of the coupler point C in the relative coordinate system$X_C\mathbf{A}Y_C$. x_0 and y_0 are the coordinates of joint O_2 in the world coordinate system XOY. θ_0 is the angle of the stationary link with respect to the X-axis. θ_2^i is the *i* th input angle corresponding to the *i* th desired point in the relative coordinate system $X_r\mathbf{O_2}Y_r$. n is the number of target points.

The positions of coupler points corresponding to the set of input angles $\{\theta_2^i\}$ in the world coordinate system XOY can be expressed as:

$$\left\{C^i(\mathrm{x})\right\} = \left\{C_x^i(x), C_y^i(x)\right\}, i = 1...n. \tag{21}$$

Angle θ_3 can be solved by the Freudenstein equation [16]. In the world coordinate system XOY, the *i* th position of the coupler point C for design variables $\boldsymbol{X}$ can be written as:

$$\begin{cases} C_x^i(x) = x_0 + r_2\cos\left(\theta_2^i + \theta_0\right) + r_{cx}\cos\left(\theta_3^i + \theta_0\right) - r_{cy}\sin\left(\theta_3^i + \theta_0\right) \\ C_y^i(x) = y_0 + r_2\sin\left(\theta_2^i + \theta_0\right) + r_{cx}\sin\left(\theta_3^i + \theta_0\right) + r_{cy}\cos\left(\theta_3^i + \theta_0\right) \end{cases} \tag{22}$$

6.2. Constraints

This application considers two constraints as follows:

Table 9
The best fitness value, mean time and mean LSLI ratio for the synthesis problem of a four-bar mechanism with different $LR^{\max}$.

Func.	jDE	$LR^{\max}=0.5$	$LR^{\max}=0.6$	$LR^{\max}=0.7$	$LR^{\max}=0.8$	$LR^{\max}=0.9$
best fitness value	4.9840E-02	4.8276E-02	4.5183E-02	4.1269E-02	3.6728E-02	2.7779E − 02
mean time (sec)	1.5969	1.9457	2.0879	2.1610	2.2396	2.3784
mean LSLI ratio	0%	23%	26%	29%	32%	35%

Table 10
Comparisons of five algorithms for the synthesis problem of a four-bar mechanism.

Parameters	CLPSO	jDE	ETLBO	LBSA	ADELI
r_1(mm)	13.908153	36.883826	37.404268	40.492953	1.153260
r_2(mm)	0.458440	0.200600	0.445328	0.309619	0.231807
r_3(mm)	7.944657	41.834985	37.575452	22.530718	44.543997
r_4(mm)	12.219031	23.367632	23.526969	26.189995	44.550351
r_{cx}(mm)	4.193310	-50.000000	20.302872	-8.154136	0.704410
r_{cy}(mm)	0.828994	5.039720	7.572097	14.942773	1.107564
x_0(mm)	-1.024549	50.000000	-0.379731	-9.648269	1.290936
y_0(mm)	-3.405335	-6.519823	22.321020	14.514138	-0.103672
θ_0(rad)	0.009567	5.650408	3.745694	2.620663	6.222545
θ_2^1(rad)	0.975326	1.444845	3.690064	4.969313	1.189289
best fitness value	9.3610E−02	4.9840E−02	4.8535E−02	4.6526E−02	**2.7779E−02**
mean	1.9967E+00	1.8399E−01	6.5862E−01	**1.1796E−01**	1.5101E−01
std.	8.1874E−01	1.1679E−01	7.8835E−01	**6.5027E−02**	1.0776E−01
mean time (sec)	**1.3636E+00**	1.5969E+00	6.3470E+00	1.3130E+00	2.3784E+00
time std.	**3.9613E−0**2	4.8458E−02	1.2339E−01	7.4491E−02	8.2202E−02

Algorithm 1
Pseudocode of DE.

Begin
Initialize *N, D,* x^{up}, x^{low}, *F, CR,* max*FES* (the max number of fitness evaluations).
Initialize the population $\boldsymbol{X}=\{x_1; ...; x_N\}$ and evaluate the fitness value of $\boldsymbol{X}$.
$FES=N$;
%Main loop
while ($FES <$ max*FES*)
  for $i=1{:}N$
    %Mutation
    Randomly choose three different indexes *ra, rb, rc*, all different from *i.*
    $v_i^{\text{child}} = x_{ra} + F(x_{rb} - x_{rc})$; ***%Yield a mutation vector*** v_i^{child}
    %Crossover
    Randomly choose a dimension j_{rand}.
    for $j=1{:}D$
      if ($v_{i,j} > x_j^{\text{up}}$) or ($v_{i,j} < x_j^{low}$)
        $v_{i,j} = (x_j^{\text{up}} - x_j^{low}) * \text{rand}(0,1) + x_j^{low}$; ***%Check boundary***
      end if
      $u_{i,j}^{\text{child}} = \begin{cases} v_{i,j}^{\text{child}} & if(\text{rand} < \text{CR}) \text{or} (j = j_{\text{rand}}) \\ x_{i,j} & \text{otherwise} \end{cases}$; ***%Generate a trial vector*** u_i^{child}
    end for *j*
    %Selection
    Evaluate the trial vector $u_i^{\text{child}} = [u_{i,1}^{\text{child}}, u_{i,2}^{\text{child}}, ..., u_{i,D}^{\text{child}}]$ with the objective function.
    $[x_i^{\text{child}}, f(x_i^{\text{child}})] = \begin{cases} [u_i^{\text{child}}, f(u_i^{\text{child}})] & if(f(u_i^{\text{child}}) < f(x_i)) \\ [x_i, f(x_i)] & \text{otherwise} \end{cases}$; %***Yield a child vector*** x_i^{child}
  end for *i*
  $FES = FES + N$;
end while

FES: the number of fitness evaluations.

6.2.1. Grashof condition

To make sure the shortest link can rotate a full 360° with respect to the neighboring link, the links' length of the four-bar linkage must satisfy the Grashof condition, which can be expressed by:

$$r_s + r_l \le r_p + r_q \tag{23}$$

where r_s is the length of the shortest link, r_l is the length of the longest link, and r_p and r_q are the lengths of the remaining two links.

In addition, the input link should be a crank in this study, i.e., the input link is the shortest link:

$$r_2 = r_s \tag{24}$$

Algorithm 2
Pseudocode of Lagrange interpolation method.

Input:
x_j^0, x_j^1, x_j^2 (three given points in the j th-dimensional space of vectorsx^0, x^1, x^2), and their fitness values$f(x^0)$, $f(x^1)$, $f(x^2)$
Output:
x_j^3(the optimum point in the j th dimension of vector x^3).
$P(x_j) = a(x_j)^2 + bx_j + c$ ***%the Lagrange interpolation polynomial***
if ($I \neq 0$)
  if ($a > 0$)
    $x_j^3 = -b/2a$;
  elseif ($a = 0$ and $b = 0$)
    $x_j^3 = (x_j^1 + x_j^2)/2 + (\text{rand} - 0.5)(x_j^1 - x_j^2)$;
  else
    Rank x_j^0, x_j^1, x_j^2 in terms of $f(x^0), f(x^1), f(x^2)$;
    $x_j^3 = x_j^{\min} + \text{rand} * c_3 * (x_j^{\min} - x_j^{\text{mid}}) + \text{rand} * c_4 * (x_j^{\min} - x_j^{\max})$;
  end if
else
  Rank x_j^0, x_j^1, x_j^2 in terms of $f(x^0), f(x^1), f(x^2)$;
  $x_j^3 = x_j^{\min} + \text{rand} * c_3 * (x_j^{\min} - x_j^{\text{mid}}) + \text{rand} * c_4 * (x_j^{\min} - x_j^{\max})$;
end if

Algorithm 3
Pseudocode of LSLI.

Input: X, D, x^{up}, x^{low}, x^{best} (the best individual), $f(x^{best})$ (fitness of x^{best}).
Output: x^{best},$f(x^{best})$.
%Local search in every dimension of neighboring x^{best} ***using*** **Algorithm 2**
for $j = 1{:}D$
  %Initialize two temporary individuals neighboring x^{best}
  $x^1 = x^{best}$;
  $x^2 = x^{best}$;
  %Initialize the potential best individual
  $x^3 = x^{best}$;
  %Reassign x^1 ***and*** x^2 ***in the j th-dimensional space***
  Reassign x_j^1 and x_j^2 in the j th-dimensional space of x^1 and x^2 by Eqs. (13) and (14).
  Evaluate temporary individuals x^1 and x^2 with Eq. (29).
  %Generate variable x_j^3 ***for*** x^3 ***using*** **Algorithm 2**
  $x_j^3 = Lagrange[x_j^{best}, x_j^1, x_j^2, f(x^{best}), f(x^1), f(x^2)]$;
  %Check boundary for variable x_j^3
  if ($x_j^3 > x_j^{up}$) or ($x_j^3 < x_j^{low}$)
    $x_j^3 = (x_j^{up} - x_j^{low}) * \text{rand}(0, 1) + x_j^{low}$;
  end if
  Evaluate individual x^3 with Eq. (29).
  %Update x^{best},$f(x^{best})$
  if ($f(x^1) < f(x^{best})$)
    Replace x^{best},$f(x^{best})$ with x^1,$f(x^1)$.
  end if
  if ($f(x^2) < f(x^{best})$)
    Replace x^{best},$f(x^{best})$ with x^2,$f(x^2)$.
  end if
  if ($f(x^3) < f(x^{best})$)
    Replace x^{best}, $f(x^{best})$ with x^3,$f(x^3)$.
  end if
end for j

6.2.2. Sequence condition

To avoid the order defect, the sequence of input angles $\{\theta_2^i\}$ should fulfil the clockwise (CW) or counterclockwise (CCW) rotation:

$$\text{CW}: \theta_2^k < \theta_2^{k-1} < \ldots < \theta_2^2 < \theta_2^1 < \theta_2^n < \ldots < \theta_2^{k+1} \tag{25}$$

$$\text{CCW}: \theta_2^k < \theta_2^{k+1} < \ldots < \theta_2^n < \theta_2^1 < \theta_2^2 < \ldots < \theta_2^{k-1} \tag{26}$$

where $\theta_2^k = \min\{\theta_2^i\}, i = 1, \ldots, n$.

6.3. Objective function

This optimization problem is to minimize the error between the coupler points and target points. The target points should be reached by the coupler point on the synthesized four-bar mechanism, which can be given in the world coordinate

Algorithm 4
Pseudocode of ADELI.

Begin
Initialize N, D, x^{up}, x^{low}, maxFEs, F, F^{low}, F^{up}, τ_1, τ_2,CR.
Initialize LR (LSLI rate), LR^{min} (the minimum of LR), LR^{max} (the maximum LR).
%Initialize population X, fitness of X and the best individual [x^{best},f(x^{best})]
for $i = 1{:}N$
  for $j = 1{:}D$
    $x_{i,j} = (x_j^{up} - x_j^{low}) * \text{rand}(0,1) + x_j^{low}$;
  end for j
end for i
Evaluate the initial population $\boldsymbol{X} = \{x_1; \ldots; x_N\}$.
Memorize the best individual (x^{best}) and its fitness (f(x^{best})).
$FEs = N; G = 1$;
%Main loop
while ($FEs < \max FEs$)
  % Adaptive argument strategy
  if ($rand(0,1) < LR_G$)
    $[x_{son}^{best}, f(x_{son}^{best})] = LocalSearch(X, D, FEs, x^{up}, x^{low}, x^{best}, f(x^{best}))$; **%**Algorithm 3
    $FEs = FEs + 3D$;
    if ($f(x_{son}^{best}) < f(x^{best})$)
      $LR_{G+1} = \min(LR_G + 1, LR^{max})$;
      Update x^{best} and$f(x^{best})$using $x_{son}^{best}$and $f(x_{son}^{best})$.
    else
      $LR_{G+1} = \max(LR_G - 1, LR^{min})$;
    end if
  end if
  for $i = 1{:}N$
    %Mutation
    Randomly pick three different indexes ra, rb, rc, all different from i.
    $F_{i,G+1} = \begin{cases} F^{low} + rand(0,1) * F^{up} & rand(0,1) < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases}$; ***%Self-adapting control parameter*** F
    $v_i^{child} = x_{ra} + F(x_{rb} - x_{rc})$; %***Generate a mutation vector*** v_i^{child}
    %Crossover
    Randomly pick a dimension j_{rand}.
    $CR_{i,G+1} = \begin{cases} rand(0,1) & rand(0,1) < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases}$; %***Self-adapting control parameter*** CR
    for $j = 1{:}D$
      if ($v_{i,j} > x_j^{up}$) or ($v_{i,j} < x_j^{low}$)
        $v_{i,j} = (x_j^{up} - x_j^{low}) * \text{rand}(0,1) + x_j^{low}$; ***%Check boundary***
      end if
      $u_{i,j}^{child} = \begin{cases} v_{i,j}^{child} & if(\text{rand}(0,1) < CR) \text{or} (j = j_{rand}) \\ x_{i,j} & \text{otherwise} \end{cases}$; %***Yield a trial vector*** u_i^{child}
    end for j
    %Selection
    Evaluate the trial vector u_i^{child}.
    $[x_i^{child}, f(x_i^{child})] = \begin{cases} [u_i^{child}, f(u_i^{child})] & if(f(u_i^{child}) < f(x_i)) \\ [x_i, f(x_i)] & \text{otherwise} \end{cases}$; %***Generate a child vector*** x_i^{child}
    end for i
    %Update population and the best individual
    $\mathbf{X} = \mathbf{X}_{child}$, $f(\mathbf{X}) = f(\mathbf{X}_{child})$, $f_{old}(x^{best}) = f(x^{best})$;
    Rank population and renew the best solution x^{best} and f(x^{best}).
  $FEs = FEs + N$; $G = G + 1$;
end while

system as follows:

$$\left\{C_d^i\right\} = \left\{C_{dx}^i, C_{dy}^i\right\}, i = 1 \ldots n. \tag{27}$$

The objective function consists of two parts. The first part of objective function is the sum of the square of the Euclidean distances between the coupler points and the corresponding target points, which can be defined as:

$$f(x) = \sum_{i=1}^{n} \left[\left(C_x^i(x) - C_{dx}^i\right)^2 + \left(C_y^i(x) - C_{dy}^i\right)^2 \right] \tag{28}$$

The second part of the objective function addresses the penalty functions. The Grashof condition and the sequence condition are introduced as the penalty functions. Finally, the objective function of the optimal problem can be expressed as:

Minimize:

$$f_{obj}(x) = \sum_{i=1}^{n} \left[\left(C_x^i(x) - C_{dx}^i\right)^2 + \left(C_y^i(x) - C_{dy}^i\right)^2 \right] + h_1(x)M_1 + h_2(x)M_2 \tag{29}$$

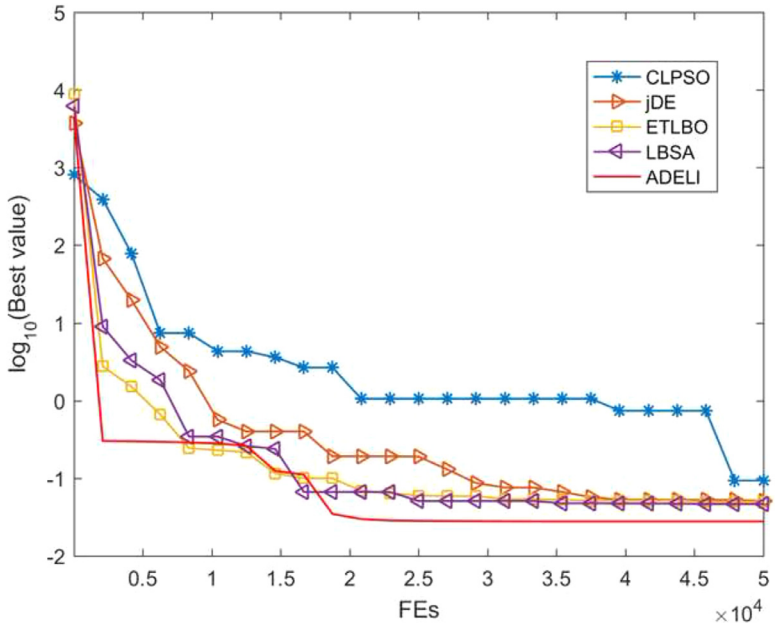

Fig. 5. Convergence of the best values of the objective function of five algorithms.

Subject to: $x_j \in [x_j^{low}, x_j^{up}], \forall x_j \in x, j = 1, ..., D$ where $h_1(x) = 1$ denotes that the Grashof condition is not satisfied, and $h_1(x) = 0$ represents that the Grashof condition is satisfied. $h_2(x) = 1$ denotes that the sequence condition is not satisfied, and $h_2(x) = 0$ represents that the sequence condition is satisfied. M_1 and M_2 are the high values to penalize the objective function; here, M_1, $M_2 = 10^4$, and D is the number of variables to be optimized. x_j^{low} and x_j^{up} are the lower and upper boundaries of the j th design variables, respectively.

6.4. Experimental settings for the synthesis problem

The mechanism synthesis problem is chosen from Cabrera [4]. This problem requires the couple point C to trace a trajectory along eighteen target points arranged in an irregular closed path with prescribed timing. Then, for this problem, the following are defined.

The design variables are:

$$x=[r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, x_0, y_0, \theta_0, \theta_2^1].$$

The target points are:

$$\left\{C_d^i\right\}=[(0.5, 1.1); (0.4, 1.1); (0.3, 1.1); (0.2, 1); (0.1, 0.9); (0.005, 0.75); (0.02, 0.6); (0, 0.5); (0, 0.4); (0.03, 0.3); (0.1, 0.25); (0.15, 0.2); (0.2, 0.3); (0.3, 0.4); (0.4, 0.5); (0.5, 0.7); (0.6, 0.9); (0.6, 1)],$$
$$\left\{\theta_2^i\right\}=\{\theta_2^1 + (i-1)\tfrac{\pi}{9}\}, i = 1, ..., 18.$$

The limits of the variables are:

$$r_1, r_2, r_3, r_4 \in [0, 50];\ r_{cx}, r_{cy}, x_0, y_0 \in [-50, 50];\ \theta_0, \theta_2^1 \in [0, 2\pi].$$

In this experiment, five algorithms, including CLPSO, jDE, ETLBO, LBSA, and ADELI, are implemented to synthesize a four-bar mechanism. Before the experiment of the synthesis of a four-bar mechanism, we conduct an experiment with five ADELIs with different LR^{max} from the set {0.5, 0.6, 0.7, 0.8, 0.9} and compare them with jDE. Considering the best fitness value, mean time, and mean LSLI ratio, the results over thirty independent runs for the four-bar synthesis problem are shown in Table 9. From the results in Table 9, we can observe that the adaptive argument mechanism of LSLI can work well by using different LSLI ratios. All five ADELIs obtain solutions with higher accuracy than jDE; moreover, when $LR^{max} = 0.9$, ADELI's solution is the most accurate. However, they require slightly more time than jDE. For the synthesis problem of a four-bar mechanism, the designer's most important goal is to ensure the solution accuracy. Hence, considering the solution accuracy, we set LR^{max} to 0.9 in the synthesis of the four-bar mechanism experiment. The max*FEs* is set to 10,000. The other parameter settings of the five algorithms are the same as those in Section 5.2.

6.5. Experimental results

The values of the best, the mean, and the standard deviations as well as the average convergence time and the standard deviations of the time achieved by the five algorithms for 30 runs are presented in Table 10. The best results are shown in boldface. According to the experimental results shown in Table 10, the best solution obtained by ADELI is the most accurate among the five algorithms. The mean and standard deviations obtained by LBSA are the best. The average convergence time and its standard deviations of CLPSO are the smallest.

The convergence graphs of the best values for this synthesis problem of the four-bar mechanism are shown in Fig. 5. It illustrates that the convergence speed of ADELI is the fastest, and the convergence accuracy of ADELI is the highest among the five contestant algorithms.

For this mechanism synthesis problem, the main aim is to minimize the error between coupler points and target points. Hence, as indicated in Table 10 and Fig. 5, ADELI is superior to the other five algorithms with respect to the convergence speed and accuracy for this optima synthesis problem of four-bar mechanisms.

7. Conclusions

In this paper, a novel algorithm, called ADELI, has been presented as an extension of standard DE, which takes advantage of both DE and the Lagrange interpolation algorithm. In ADELI, a local search technique with Lagrange interpolation is applied to accelerate the convergence, and an adaptive argument strategy is proposed to perform LSLI near the best individual to discourage becoming trapped in local optima. Moreover, a self-adaptive method is introduced for controlling the parameter selection during the evolutionary process.

The performance of ADELI was tested on numerical benchmark functions in the CEC 2014 sets and compared to ten other EAs. The results show that ADELI with the adaptive local search technique is significantly better than the other ten EAs in most cases. For the application of mechanism synthesis, the optimization results also indicate that ADELI has the most accurate solution and the fastest convergence speed among the compared algorithms.

The proposed local search scheme with an adaptive argument strategy can be inserted into any other EAs to enhance their local exploitation capability. Future works will focus on extending ADELI's application domain to handle more practical optimization problems.

Acknowledgments

The authors would like to thank the reviewers for their suggestions as well as the constructive comments for this article. This work was supported by the National Natural Science Foundation of China (Grant no. U1708254).

Authors' statement

The authors have no competing interests to declare.

References

[1] I. Ahmadianfar, A. Samadi-Koucheksaraee, O. Bozorg-Haddad, Extracting optimal policies of hydropower multi-reservoir systems utilizing enhanced differential evolution algorithm, Water Resour. Manag. (10) (2017) 1–23.
[2] G.E. Box, J.S. Hunter, W.G. Hunter, Statistics For experimenters: design, innovation, and Discovery, Wiley-Interscience, New York, 2005.
[3] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[4] J. Cabrera, A. Simon, M. Prado, Optimal synthesis of mechanisms with genetic algorithms, Mech. Mach. Theory 37 (10) (2002) 1165–1177.
[5] J. Chaturvedi, Adaptive quantum inspired genetic algorithm for combinatorial optimization problems, Int. J. Comput. Appl. 107 (4) (2014).
[6] D. Chen, R. Lu, F. Zou, S. Li, P. Wang, A learning and niching based backtracking search optimisation algorithm and its applications in global optimisation and ANN training,, Neurocomputing 266 (2017) 579–594.
[7] D. Chen, F. Zou, R. Lu, P. Wang, Learning backtracking search optimisation algorithm and its application, Inf. Sci. 376 (2017) 71–94.
[8] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Appl. Math. Comput. 219 (15) (2013) 8121–8144.
[9] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution–an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.
[10] S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[11] R. Dash, P.K. Dash, R. Bisoi, A self adaptive differential harmony search based optimized extreme learning machine for financial time series prediction, Swarm Evol. Comput. 19 (2014) 25–42.
[12] M. Dorigo, Optimization, learning and natural algorithms Ph. D. Thesis, Politecnico di Milano, Italy, 1992.
[13] H. Duan, Q. Luo, Adaptive backtracking search algorithm for induction magnetometer optimization, IEEE Trans. Magn. 50 (12) (2014) 1–6.
[14] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive and social experience in particle swarm optimization through differential evolution: a hybrid approach, Inf. Sci. 216 (2012) 50–92.
[15] H.Y. Fan, J. Lampinen, A Trigonometric Mutation Operation to Differential Evolution, J. Global Optim. 27 (1) (2003) 105–129.
[16] F. Freudenstein, An analytical approach to the design of four-link mechanisms, Trans. ASME 76 (3) (1954) 483–492.
[17] Y. Fu, M. Ding, C. Zhou, Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV, IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. 42 (2) (2012) 511–526.
[18] A.H. Gandomi, A.H. Alavi, Krill herd: A new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4831–4845.
[19] H. Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Appl. Math. Comput. 274 (2016) 292–305.
[20] S.-M. Guo, C.-C. Yang, Enhancing differential evolution utilizing eigenvector-based crossover operator, IEEE Trans. Evol. Comput. 19 (1) (2015) 31–49.
[21] K.-H. Han, J.-H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, IEEE Trans. Evol. Comput. 6 (6) (2002) 580–593.

[22] J.H. Holland, Adaptation in Natural and Artificial Systems. An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence, University of Michigan Press, Ann Arbor, MI, 1975.
[23] Z. Kai, S. Jinchun, N. Ke, L. Song, Lagrange interpolation learning particle swarm optimization, Plos One 11 (4) (2016) e0154191.
[24] J. Kennedy, R. Eberhart, in: Particle swarm optimization [a] proceedings of the ieee international conference on neural networks [c] piscataway, NJ, USA: IEEE, 1995.
[25] J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2013).
[26] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[27] M. Nasir, S. Das, D. Maity, S. Sengupta, U. Halder, P.N. Suganthan, A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization, Inf. Sci. 209 (2012) 16–36.
[28] N. Noman, H. Iba, Enhancing differential evolution performance with local search for high dimensional function optimization, in: Proceedings of the 7th annual conference on Genetic and evolutionary computation, 2005, pp. 967–974.
[29] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. Evol. Comput. 12 (1) (2008) 107–125.
[30] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[31] R. Rao, V. Patel, An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems, Int. J. Indust. Eng. Comput. 3 (4) (2012) 535–560.
[32] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput. aided Des. 43 (3) (2011) 303–315.
[33] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: International Conference on Evolutionary Programming, 1998, pp. 591–600.
[34] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, 3, 1999, pp. 1945–1950.
[35] H. Shieh, C. Kuo, C. Chiang, Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification, Appl. Math. Comput. 218 (8) (2011) 4365–4383.
[36] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, Eur. J. Oper. Res. 185 (3) (2008) 1155–1173.
[37] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[38] M.R. Tanweer, S. Suresh, N. Sundararajan, Dynamic mentoring and self-regulation based particle swarm optimization algorithm for solving complex real-world optimization problems, Inf. Sci. 326 (2016) 1–24.
[39] V. Tirronen, F. Neri, T. Rossi, Enhancing differential evolution frameworks by scale factor local search-part i„ in: Evolutionary Computation, 2009. CEC'09. IEEE Congress on, 2009, pp. 94–101.
[40] A. Trivedi, D. Srinivasan, S. Biswas, T. Reindl, A genetic algorithm–differential evolution based hybrid framework: case study on unit commitment scheduling problem, Inf. Sci. 354 (2016) 275–300.
[41] L. Wang, Y. Zhong, Y. Yin, W. Zhao, B. Wang, Y. Xu, A hybrid backtracking search optimization algorithm with differential evolution, Math. Probl. Eng. 2015 (2015).
[42] W. Xie, W. Yu, X. Zou, Diversity-maintained differential evolution embedded with gradient-based local search, Soft Comput. 17 (8) (2013) 1511–1535.
[43] X. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2010.
[44] X.S. Yang, S. Deb, Cuckoo search via levy flights, Mathematics (2010) 210–214.
[45] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[46] S.Z. Zhao, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with harmony search, Expert Syst. . Appl. 38 (4) (2011) 3735–3742.
[47] F. Zou, L. Wang, X. Hei, D. Chen, D. Yang, Teaching–learning-based optimization with dynamic group strategy for global optimization, Inf. Sci. 273 (2014) 112–131.