Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 1589303, 24 pages
https://doi.org/10.1155/2019/1589303

Research Article

An Enhanced Lightning Attachment Procedure Optimization with Quasi-Opposition-Based Learning and Dimensional Search Strategies

Tongyi Zheng and Weili Luo

School of Civil Engineering, Guangzhou University, Guangzhou, China

Correspondence should be addressed to Weili Luo; wlluo@gzhu.edu.cn

Received 15 February 2019; Revised 15 June 2019; Accepted 17 July 2019; Published 1 August 2019

Academic Editor: Bruce J. MacLennan

Copyright © 2019 Tongyi Zheng and Weili Luo. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Lightning attachment procedure optimization (LAPO) is a new global optimization algorithm inspired by the attachment procedure of lightning in nature. However, similar to other metaheuristic algorithms, LAPO also has its own disadvantages. To obtain better global searching ability, an enhanced version of LAPO called ELAPO has been proposed in this paper. A quasi-opposition-based learning strategy is incorporated to improve both exploration and exploitation abilities by considering an estimate and its opposite simultaneously. Moreover, a dimensional search enhancement strategy is proposed to intensify the exploitation ability of the algorithm. 32 benchmark functions including unimodal, multimodal, and CEC 2014 functions are utilized to test the effectiveness of the proposed algorithm. Numerical results indicate that ELAPO can provide better or competitive performance compared with the basic LAPO and other five state-of-the-art optimization algorithms.

1. Introduction

Optimization problems can be found in many engineering application domains and scientific fields which have a complex and nonlinear nature. It is usually difficult to solve these optimization problems using classical mathematical methods since such methods are often inefficient and have a requirement of strong math assumptions. Due to the limitations of classical approaches, many natural-inspired stochastic optimization algorithms have been proposed to conduct global optimization problems in the last two decades. Such optimization algorithms were commonly simple and easy to implement, and these features make the possibility to solve highly complex optimization problems. These metaheuristics can be roughly classified into three categories: evolutionary algorithms, swarm intelligence, and physical-based algorithms.

Evolutionary algorithms are generic population-based metaheuristics, which imitate the evolutionary behavior of biology in nature such as reproduction, mutation, recombination, and selection. The first generation starts with randomly initialized solutions and further evolves over successive generations. The best individual among the whole population in the final evolution is considered to be the optimization solution. Some of the popular evolutionary algorithms are genetic algorithm (GA) [1], genetic programming (GP) [2], evolution strategy (ES) [3], differential evolution (DE) algorithm [4], and biogeography-based optimizer (BBO) [5].

Swarm intelligence algorithms mimic the collective behavior of swarms, herds, schools, or flocks of creatures in nature, which interact with each other and utilize full information about their environment with the progress of algorithm. For example, honey bees are capable of guaranteeing the survival of a colony without any external guidance. In other words, no one tells honey bees how and where to find food sources; instead, they cooperatively seek the food sources even that is located far away from their nests. In this category, particle swarm optimization (PSO) [6], ant colony optimization (ACO) [7], and artificial bee colony algorithm (ABC) [8] can be regarded as representative algorithms. Some other popular swarm intelligence algorithms are firefly mating algorithm (FMA) [9], shuffled

frog leaping algorithm (SFLA) [10], bee collecting pollen algorithm (BCPA) [11], cuckoo search (CS) algorithm [12], dolphin partner optimization (DPO) [13], bat-inspired algorithm (BA) [14], firefly algorithm (FA) [15], and hunting search (HUS) algorithm [16]. Some of the recent swarm intelligence algorithms are fruit fly optimization algorithm (FOA) [17], dragonfly algorithm (DA) [18], artificial algae algorithm (AAA) [19], ant lion optimizer (ALO) [20], shark smell optimization algorithm (DSOA) [21], whale optimization algorithm (WOA) [22], crow search algorithm (CSA) [23], grasshopper optimization algorithm (GOA) [24], mouth brooding fish algorithm (MBFA) [25], spotted hyena optimizer (SHO) [26], butterfly-inspired algorithm (BFA) [27], squirrel search algorithm (SSA) [28], Andean condor algorithm (ACA) [29], and pity beetle algorithm (PBA) [30].

The third category is physical-based algorithms which are based on the basic physical laws such as gravitational force, electromagnetic force, and inertia force. Some of the prevailing algorithms of this category are simulated annealing (SA) [31], gravitational search algorithm (GSA) [32], big-bang big-crunch (BBBC) algorithm [33], charged system search (CSS) [34], black hole (BH) algorithm [35], central force optimization (CFO) [36], small-world optimization algorithm (SWOA) [37], artificial chemical reaction optimization algorithm (ACROA) [38], ray optimization (RO) algorithm [39], galaxy-based search algorithm (GbSA) [40], and curved space optimization (CSO) [41], gravitational search algorithm (GSA) [32], and multiverse optimizer (MVO) [42].

Regardless of the difference among the three categories of algorithms, a common point lies in that besides tuning of common control parameters such as population size and number of generations, the metaheuristic algorithms necessitate tuning of algorithm-specific parameters during the course of optimization. For instance, GA requires tuning of cross-over probability, mutation probability, and selection operator [43]; SA requires tuning of initial temperature and cooling rate [31]; PSO requires tuning of inertia weight and learning factors [6]. The improper tuning of these parameters either increases the computational cost or leads to the local optimal solution.

Recently, a new physical-based metaheuristic algorithm named lightning attachment procedure optimization (LAPO) [44] was proposed, which does not require tuning of any algorithm-specific parameters. Instead, an average value of all solutions was employed to adjust the lightning jump behavior of moving towards or away from a jumping point (or position) in a self-adaptive manner. This is an important reason that LAPO is not easily stuck in the local optimal solution and has a good exploration and exploitation abilities. LAPO has already proved its superiority in solving a number of constrained numerical optimization problems [44].

In this paper, an enhanced lightning attachment procedure optimization, namely, ELAPO is developed to increase the convergence speed during the search process of LAPO while maintaining the key feature of the LAPO as free from algorithm-specific parameters tuning. In ELAPO, a concept of opposition-based learning (OBL) is incorporated for enhancing the searching ability of metaheuristic algorithms. The motivation is that the current estimates and their corresponding opposites are considered simultaneously to find the better solutions, thereby enabling the algorithm to explore a large region of the search space in every generation. This concept was found to be effective in improving the performance of well-known optimization algorithms such as genetic algorithms (GA) [45], differential evolution (DE) [46, 47], particle swarm optimization (PSO) [48, 49], biogeography-based optimization (BBO) [50, 51], harmony search (HS) algorithm [52, 53], gravitational search optimization (GSO) [54, 55], group search algorithm (GSA) [56, 57], and artificial bee colony (ABC) [58]. Meanwhile, a dimensional search strategy is proposed to intensively exploit a local search for each variable of the best solution in each iteration, thus resulting in a higher quality of solution at the end of iteration and strengthening the exploitation of the algorithm. To evaluate the effectiveness of the proposed algorithms, ELAPO is applied to 32 benchmark functions and compared with the basic LAPO and six representative swarm intelligence algorithms (SSA [28], Jaya [59], IBB-BC [60], ODE1 [61], and ALO [20]). The effectiveness of the two strategies is also discussed.

The rest of this paper is organized as follows: Section 2 briefly recapitulates the basic LAPO. Next, the proposed ELAPO is presented in a detailed way in Section 3. Numerical comparisons are illustrated in Section 4. Finally, Section 5 gives the concluding remarks.

2. Basic Algorithm

LAPO is a new nature-inspired global optimization, which mimics the lightning attachment procedure including the downward leader movement and the upward leader propagation. The lightning is a sudden electrostatic discharge occurring between electrically charged regions of a cloud, which moves toward or away from the ground in a stepwise movement. After each step, the downward leader stops and then moves to a randomly selected potential point that may have higher value of electrical field. The upward leader starts from sharp points and goes towards the downward leader. The branch fading feature of lightning is taken effect when the charge of a branch is lower than a critical value. In the case where the two leaders join together, a final strike occurs and the charge of the cloud is neutralized.

2.1. Parameters and Initialization of Test Points. Main parameters of the LAPO consist of the maximum number of iterations $\text{Iter}_{\max}$, the number of test points $N\text{pop}$, the number of decision variables n, and the upper and lower bounds for decision variable $X_{\max}$ and $X_{\min}$. These parameters are given at the beginning of the algorithm. Similar to other nature-inspired optimization algorithms, an initial population is required. Each population is regarded as a test point in the feasible search space, which could be an emitting point of the downward or upward leader. The test points are randomly initialized as follows:

$$X_{i,j} = X_{\min} + \text{rand}(\,) * (X_{\max} - X_{\min}),$$
$$i = 1, 2, \ldots, N\text{pop}, \quad j = 1, 2, \ldots, n, \tag{1}$$

where rand( ) is a uniformly distributed random number in the range [0, 1]. The electric field (i.e., fitness value) $f = (f_1, f_2, \ldots, f_{N\text{pop}})$ of each test point is calculated based on the objective function:

$$f_i = \text{obj}\left(X_{i,1}, X_{i,2}, \ldots, X_{i,n}\right), \quad i = 1, 2, \ldots, N\text{pop}. \tag{2}$$

2.2. Downward Leader Movement toward the Ground. In this phase, all the test points are considered as the downward leader and move down towards the ground. The average value of all test points and its corresponding fitness value are calculated as follows:

$$X_{\text{ave}} = \text{mean}\left(X_{i,j}\right), \tag{3}$$

$$f_{\text{ave}} = \text{obj}\left(X_{\text{ave}}\right). \tag{4}$$

Given the fact that the lightning has a random behavior, for test point i, a random point k is selected among the population ($i \neq k$), and the new test point is updated based on the following rules: (i) if the electric field of point k is higher than the average electric field, then

$$X_{i,j}^{\text{new}} = X_{i,j} + \text{rand}( ) * \left(X_{\text{ave}} - \text{rand}( ) * X_{k,j}\right), \tag{5}$$

and (ii) if the electric field of point k is lower than the average electric field, then

$$X_{i,j}^{\text{new}} = X_{i,j} - \text{rand}( ) * \left(X_{\text{ave}} - \text{rand}( ) * X_{k,j}\right). \tag{6}$$

If the electric field of the new test point is better than the old one, the branch sustains; otherwise, it fades. This feature is mathematically formulated as

$$X_{i,j} = \begin{cases} X_{i,j}^{\text{new}}, & \text{if } f\left(X_{i,j}^{\text{new}}\right) < f\left(X_{i,j}\right), \\ X_{i,j}, & \text{otherwise.} \end{cases} \tag{7}$$

2.3. Upward Leader Movement. In the upward movement phase, all the test points are considered as the upward leader towards the cloud. The new test points are generated as follows:

$$X_{i,j}^{\text{new}} = X_{i,j} + \text{rand}( ) * S * \left(X_{\text{best}} - X_{\text{worst}}\right), \tag{8}$$

where X_{best} and X_{worst} are the best and the worst solutions of the population and S is an exponent factor that is a function of the number of iterations Iter and the maximum number of iterations Iter_{max}:

$$S = 1 - \left(\frac{\text{Iter}}{\text{Iter}_{\text{max}}}\right) * \exp\left(\frac{\text{Iter}}{\text{Iter}_{\text{max}}}\right). \tag{9}$$

From a computational point of view, this iteration-dependent exponent factor is important for the balance of exploration and exploitation capabilities of the algorithm. Similar to the downward movement, the branch fading feature also occurs in this phase.

2.4. Enhancement of the Performance. In order to enhance the performance of LAPO, in each iteration, the worst test point is replaced by the average test point if the fitness of the former is worse than the latter:

$$X_{\text{worst}} = X_{\text{ave}}, \quad \text{if } f_{\text{ave}} < f\left(X_{\text{worst}}\right). \tag{10}$$

2.5. Stopping Criterion. The algorithm terminates if the maximum number of iterations is satisfied. Otherwise, the procedures of downward and upward leader movements and of performance enhancement are repeated.

2.6. Procedure of the Basic LAPO. The complete computational procedure of the basic LAPO is provided in Algorithm 1.

3. The Enhanced Lightning Attachment Procedure Optimization

The enhanced lightning attachment procedure optimization (ELAPO) is presented in this section. Two main strategies exist in the ELAPO. First, a quasi-opposition-based learning strategy is developed and employed randomly to diversify the population. Second, the dimensional search strategy is proposed to improve the quality of the best solution in each iteration. The key ideas behind ELAPO are illustrated as follows.

3.1. Quasi-Opposition-Based Learning. In order to prevent the proposed algorithm from being trapped in local optimal solutions, a monitoring condition is introduced and checked in each iteration. Following steps are involved. First, a distance constant between the average test point and the best test point is calculated:

$$D_{\text{c}} = \sqrt{\sum_{j=1}^{n} \left(X_{\text{ave},j} - X_{\text{best},j}\right)^2}. \tag{11}$$

Second, the minimum value of the distance constant condition is computed:

$$D_{\text{cmin}} = \frac{15}{10^{\left(\text{Iter}/\left(\text{Iter}_{\text{max}}\right)\right)}}, \tag{12}$$

and the monitoring condition is then checked. If $D_{\text{c}} < D_{\text{cmin}}$, the concept of opposition-based learning is employed to further diversify the population and improve the convergence rage of the algorithm. In the strategy, a portion of test points is randomly selected, based on which the corresponding opposite test points are generated and both are considered at the same time. Then, the fitness of the original test points and the quasi-opposite test points are calculated and ranked in a descending order, from which the first Npop solutions are selected for proceeding the downward leader movement and the upward leader movement. In order to maintain the stochastic nature of ELAPO, a quasi-opposite solution is randomly generated between the center of the search space CS and the mirror point of the corresponding test point MP:

(1) Set $\text{Iter}_{\max}$, $N\text{pop}$, n, $X_{\max}$ and $X_{\min}$
(2) Randomly initialize the test points
(3) $X_{i,j} = X_{\min} + \text{rand}(\,) * (X_{\max} - X_{\min}), \quad i = 1, 2, \ldots, N\text{pop}, \ j = 1, 2, \ldots, n$
(4) Calculate fitness value
(5) $f_i = \text{obj}(X_{i,1}, X_{i,2}, \ldots, X_{i,n}), \quad i = 1, 2, \ldots, N\text{pop}$
(6) **while** $\text{Iter} < \text{Iter}_{\max}$
(7) Calculate average value of all test points and its fitness value
(8) $X_{\text{ave}} = \text{mean}(X_{i,j})$
(9) $f_{\text{ave}} = \text{obj}(X_{\text{ave}})$
(10) **if** $f_{\text{ave}} < f(X_{\text{worst}})$
(11) $X_{\text{worst}} = X_{\text{ave}}$
(12) **end**
(13) Downward leader movement toward the ground
(14) **for** $i = 1: N\text{pop}$
(15) randomly select $X_{k,j}$ $(X_{k,j} \neq X_{i,j})$
(16) **if** $f_{\text{ave}} < f(X_{k,j})$
(17) $X_{i,j}^{\text{new}} = X_{i,j} + \text{rand}(\,) * (X_{\text{ave}} - \text{rand}(\,) * X_{k,j})$
(18) **else**
(19) $X_{i,j}^{\text{new}} = X_{i,j} - \text{rand}(\,) * (X_{\text{ave}} - \text{rand}(\,) * X_{k,j})$
(20) **end**
(21) Calculate fitness value of new test points
(22) **if** $f(X_{i,j}^{\text{new}}) < f(X_{i,j})$
(23) $X_{i,j} = X_{i,j}^{\text{new}}$
(24) **end**
(25) **end**
(26) Upward leader movement
(27) **for** $i = 1: N\text{pop}$
(28) $S = 1 - (\text{Iter}/\text{Iter}_{\max}) * \exp(\text{Iter}/\text{Iter}_{\max})$
(29) $X_{i,j}^{\text{new}} = X_{i,j} + \text{rand}(\,) * S * (X_{\text{best}} - X_{\text{worst}})$
(30) **if** $f(X_{i,j}^{\text{new}}) < f(X_{i,j})$
(31) $X_{i,j} = X_{i,j}^{\text{new}}$
(32) **end**
(33) **end**
(34) Iter = Iter + 1
(35) **end**

Algorithm 1: Pseudocode of basic LAPO.

$$X_{i,j}^{q} = \begin{cases} \text{CS} + \text{rand}(\,) * (\text{MP} - \text{CS}), & \text{if MP} > \text{CS}, \\ \text{MP} + \text{rand}(\,) * (\text{CS} - \text{MP}), & \text{otherwise}, \end{cases} \quad i = 1, 2, \ldots, Nq,$$

$$\text{CS} = \frac{X_{\max} + X_{\min}}{2},$$

$$\text{MP} = X_{\max} + X_{\min} - X_{i,j}, \tag{13}$$

where Nq is the number of randomly chosen test points for the generation of opposite test points, and it is set to be 5 in this paper.

3.2. Enhancing Dimensional Search. During the search process of the basic LAPO, all dimensions of each test point are updated simultaneously after each iteration. In other words, different variables in each dimension are dependent. However, this procedure has one obvious drawback: the change in one dimensional variable may cause negative impacts on other dimensional variables, thereby leading to poor convergence performance in each dimension. In order to enhance the dimensional search for each variable, the following four steps are carried out in each iteration: (a) find the best test point, (b) generate one new solution based on the best test point in a way that the value of one variable is revised while the rest of variables are preserved, (c) compare fitness values of the new-generated solution with the old solution, and reserve the better one, and (d) repeat steps (b) and (c) for other dimensional variables. The new-generated solution is produced according the following rule:

$$X_{\text{best},j}^{\text{new}} = X_{\text{best},j} + \text{rand}(\,) * S * \left(X_{\text{best},j} - X_{\text{worst},j}\right), \quad j = 1, 2, \ldots, n. \tag{14}$$

3.3. Procedure of ELAPO. The complete computational procedure of the enhanced ELAPO is provided in Algorithm 2.

4. Experimental Results and Analysis

In this section, the performance of ELPAO is evaluated by means of 32 different benchmark functions and the results

(1) Set $\text{Iter}_{\max}$, $N\text{pop}$, Nq, n, $X_{\max}$ and $X_{\min}$
(2) Randomly initialize the test points
(3) $X_{i,j} = X_{\min} + \text{rand}(\,) * (X_{\max} - X_{\min}), \quad i = 1, 2, \ldots, N\text{pop}, \ j = 1, 2, \ldots, n$
(4) Calculate fitness value
(5) $f_i = \text{obj}(X_{i,1}, X_{i,2}, \ldots, X_{i,n}), \quad i = 1, 2, \ldots, N\text{pop}$
(6) **while** $\text{Iter} < \text{Iter}_{\max}$
(7) Calculate average value of all test points and its fitness value
(8) $X_{\text{ave}} = \text{mean}(X_{i,j})$
(9) $f_{\text{ave}} = \text{obj}(X_{\text{ave}})$
(10) **if** $f_{\text{ave}} < f(X_{\text{worst}})$
(11) $X_{\text{worst}} = X_{\text{ave}}$
(12) **end**
(13) Generate quasi-opposite test points
(14) $D_c = \sqrt{\sum_{j=1}^{n} (X_{\text{ave},j} - X_{\text{best},j})^2}$, $D_{\text{cmin}} = 15/(10^{\text{Iter}/(\text{Iter}_{\max})})$
(15) **if** $D_c < D_{\text{cmin}}$
(16) $\text{CS} = (X_{\max} + X_{\min})/2$; $\text{MP} = X_{\max} + X_{\min} - X_{i,j}$
(17) $X_{i,j}^{q} = \begin{cases} \text{CS} + \text{rand}(\,) * (\text{MP} - \text{CS}), & \text{if MP} > \text{CS}, \\ \text{MP} + \text{rand}(\,) * (\text{CS} - \text{MP}), & \text{otherwise}, \end{cases} \quad i = 1, 2, \ldots, Nq,$
(18) **end**
(19) Select good solutions from the original test points and the quasi-opposite test points
(20) Downward leader movement toward the ground
(21) **for** $i = 1{:}\, N\text{pop}$
(22) randomly select $X_{k,j}$ $(X_{k,j} \neq X_{i,j})$
(23) **if** $f_{\text{ave}} < f(X_{k,j})$
(24) $X_{i,j}^{\text{new}} = X_{i,j} + \text{rand}(\,) * (X_{\text{ave}} - \text{rand}(\,) * X_{k,j})$
(25) **else**
(26) $X_{i,j}^{\text{new}} = X_{i,j} - \text{rand}(\,) * (X_{\text{ave}} - \text{rand}(\,) * X_{k,j})$
(27) **end**
(28) Calculate fitness value of new test points
(28) **if** $f(X_{i,j}^{\text{new}}) < f(X_{i,j})$
(29) $X_{i,j} = X_{i,j}^{\text{new}}$
(30) **end**
(31) **end**
(32) Upward leader movement
(33) **for** $i = 1{:}\, N\text{pop}$
(34) $S = (1 - (\text{Iter}/\text{Iter}_{\max})) * \exp(\text{Iter}/\text{Iter}_{\max})$
(35) $X_{i,j}^{\text{new}} = X_{i,j} + \text{rand}(\,) * S * (X_{\text{best}} - X_{\text{worst}})$
(36) **if** $f(X_{i,j}^{\text{new}}) < f(X_{i,j})$
(37) $X_{i,j} = X_{i,j}^{\text{new}}$
(38) **end**
(39) **end**
(40) Enhance intensive dimensional search
(41) Find X_{best}, f_{best}
(42) **for** $j = 1{:}\, n$
(43) $X_{\text{best},j}^{\text{new}} = X_{\text{best},j} + \text{rand}(\,) * S * (X_{\text{best},j} - X_{\text{worst},j}), \quad j = 1, 2, \ldots, n$
(44) Calculate fitness value of the new solution
(45) $f_{\text{best}}^{\text{new}} = f(X_{\text{best},1}, X_{\text{best},2}, \ldots, X_{\text{best},j}^{\text{new}}, \ldots, X_{\text{best},n})$
(46) **if** $f_{\text{best}}^{\text{new}} < f_{\text{best}}$
(47) $X_{\text{best},j} = X_{\text{best},j}^{\text{new}}$
(48) $f_{\text{best}} = f_{\text{best}}^{\text{new}}$
(49) **end**
(50) **end**
(51) $\text{Iter} = \text{Iter} + 1$
(52) **End**

Algorithm 2: Pseudocode of ELAPO.

Table 1: Unimodal benchmark functions.

Function	n	Range	$F_{\min}$
$F1(x) = \sum_{i=1}^{n} i x_i^2$	30, 100	$[-10, 10]$	0
$F2(x) = \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2 + (x_1 - 1)^2$	30, 100	$[-10, 10]$	0
$F3(x) = -\exp(-0.5\sum_{i=1}^{n} x_i^2)$	30, 100	$[-1, 1]$	-1
$F4(x) = \sum_{i=1}^{n} (10^6)^{((i-1)/(n-1))} x_i^2$	30, 100	$[-100, 100]$	0
$F5(x) = \sum_{i=1}^{n} i x_i^4 + \text{rand}()$	30, 100	$[-1.28, 1.28]$	0
$F6(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30, 100	$[-30, 30]$	0
$F7(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j^2)$	30, 100	$[-100, 100]$	0
$F8(x) = \max\{\lvert x_i \rvert, \quad 1 \le i \le n\}$	30, 100	$[-100, 100]$	0
$F9(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30, 100	$[-10, 10]$	0
$F10(x) = \sum_{i=1}^{n} x_i^2$	30, 100	$[-100, 100]$	0
$F11(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	30, 100	$[-1, 1]$	0

Table 2: Multimodal benchmark functions.

Function	n	Range	$F_{\min}$
$F12(x) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - \exp(1/n\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + \exp(1)$	30,100	$[-32, 32]$	0
$F13(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	30,100	$[-10, 10]$	0
$F14(x) = f_s(x_1, x_2) + \ldots + f_s(x_n, x_1),\ f_s(x, y) = (x^2 + y^2)^{0.25}[\sin^2(50(x^2 + y^2)^{0.1}) + 1]$	30,100	$[-100, 100]$	0
$F15(x) = f_s(x_1, x_2) + \ldots + f_s(x_n, x_1), f_s(x, y)$ $= 0.5((\sin^2(\sqrt{x^2 + y^2}) - 0.5)/(1 + 0.001(x^2 + y^2))^2)$	30,100	$[-100, 100]$	0
$F16(x) = \pi/n\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4),\ y_i = 1 + (1/4)(x_i + 1), u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i > a. \end{cases}$	30,100	$[-50, 50]$	0
$F17(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1$	30,100	$[-100, 100]$	0
$F18(x) = -\sum_{i=1}^{n-1}(\exp(-(x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1})/8) * \cos(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}}))$	30,100	$[-5, 5]$	$1 - n$
$F19(x) = \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	30,100	$[-n^2, n^2]$	$(n(n+4)(n-1))/-6$
$F20(x) = \sum_{i=2}^{n-1}(0.5 + (\sin^2(\sqrt{100x_i^2 + x_{i+1}^2}) - 0.5))/((1 + 0.001(x_i^2 - 2x_i x_{i-1} + x_{i-1}^2))^2)$	30,100	$[-100, 100]$	0
$F21(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	30,100	$[-5.12, 5.12]$	0
$F22(x) = \sum_{i=1}^{n}[y_i^2 - 10\cos(2\pi y_i) + 10], \quad y_i = \begin{cases} x_i, & \lvert x_i \rvert < 0.5, \\ \text{round}(2x_i)/2, & \lvert x_i \rvert < 0.5. \end{cases}$	30,100	$[-5.12, 5.12]$	0
$F23(x) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	30,100	$[-100, 100]$	0
$F24(x) = \sum_{i=1}^{n}\{\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(x_i + 0.5))]\} - n\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k 0.5)]$	30,100	$[-0.5, 0.5]$	0
$F25(x) = \sum_{k=1}^{n}\sum_{j=1}^{n}((y_{jk}^2/4000) - \cos(y_{jk}) + 1), \quad y_{jk} = 100(x_k - x_j^2)^2 + (1 - x_j^2)^2$	30,100	$[-100, 100]$	0

Table 3: CEC 2014 benchmark functions.

Function	n	Range	$F_{\min}$
F26 (CEC1: rotated high-conditioned elliptic function)	30, 100	$[-100, 100]$	100
F27 (CEC2: rotated bent cigar function)	30, 100	$[-100, 100]$	200
F28 (CEC4: shifted and rotated Rosenbrock's function)	30, 100	$[-100, 100]$	400
F29 (CEC17: hybrid function 1)	30, 100	$[-100, 100]$	1700
F30 (CEC23: composition function 1)	30, 100	$[-100, 100]$	2300
F31 (CEC24: composition function 2)	30, 100	$[-100, 100]$	2400
F32 (CEC25: composition function 3)	30, 100	$[-100, 100]$	2500

are compared to those of several state-of-the art meta-heuristic optimization algorithms. The benchmark functions are listed in Tables 1–3, among which F1–F11 are unimodal functions, F13–F25 belong to multimodal functions, and F26–F32 are composite functions provided by IEEE CEC 2014 special section [62]. In these tables, n refers to

dimension of functions, Range donates the search space, and Fmin is the true optimal value of the test functions. Two kinds of dimension ($n = 30$, and 100) are chosen in order to evaluate the capability of the proposed algorithm for solving different scale test functions.

Six metaheuristic optimization algorithms are utilized in this section as a comparison with the proposed algorithm, including the basic LAPO, squirrel search algorithm (SSA) [28], Jaya [59], improved big bang-big crunch algorithm (IBB-BC) [60], opposition-based differential evolution algorithm (ODE1) [61], and ant lion optimizer (ALO) [20]. The population size and the maximal iteration number are set to be 50 and 1000, respectively. The same set of initial random populations is used to evaluate different algorithms. The error value, defined as $f(x) - F_{\min}$, is recorded for the solution x, where $f(x)$ is the optimal fitness value of the function calculated by the algorithms. The widely used parametric settings of all algorithms are listed in Table 4. Each algorithm is applied on the test functions in 10 independent runs. The average and standard deviation of the error values over all independent runs is calculated. Meanwhile, all algorithms are compared in terms of convergence behavior with different curves (Figures 1–6). In addition, the effectiveness of each strategy is tested.

4.1. Experimental Test 1: Unimodal Functions. Unimodal benchmark functions have one global optimal solution, and they are commonly used for evaluating the exploitation ability of optimization algorithms. Tables 5 and 6 list the statistical results (the mean error and standard deviation) by different algorithms through 10 independent runs at $n = 30$ and 100, respectively. In these tables, the best values are highlighted in bold. It is obvious from the results that ELAPO has an extremely high level of accuracy and convergence precision for most of unimodal functions in comparison to other six counterpart algorithms. Taken F10 as an example, ELAPO can reach a mean error level of $10E-195$ with zero standard deviation at $n = 30$, while the accuracy of the rest algorithms is ranked in an order of LAPO ($10E-27$), ODE1 ($10E-11$), ALO ($10E-7$), SSA ($10E-6$), IBB-BC ($10E-4$), and Jaya ($10E-2$). It is also found that ELAPO is able to achieve the true minimal value of F3 and F11, while the rest of algorithms fail to obtain the same level of accuracy except for LAPO on F3. The increase in the number of dimensions seems to not affect the outstanding accuracy of ELAPO in comparison to other algorithms, though the accuracy of all algorithms tends to decrease.

Figures 1 and 2 show the convergence behaviors of some test functions for ELAPO and its competitors at $n = 30$ and 100, respectively. As can be seem from these figures, for most of test functions, ELAPO dramatically outperforms its competitors in terms of both convergence rate and precision. For F5 and F6, the convergence performance of ELAPO is still the best, though LAPO tends to have similar behaviors, and the difference between ELAPO and the rest five algorithms seems to be not very significant. Such excellent performance of ELAPO may be due to the introduction of quasi-opposition-based learning strategy as well as the dimensional search strategy.

Table 4: Parameter setting for the involved algorithms.

Algorithm	Parameter
ELAPO	—
LAPO	—
SSA	$G_c = 1.9$, $sf = 18$, $P_{dp} = 0.01$, $N_{fs} = 4$
Jaya	—
IBB-BC	$\gamma = 0.2$, $\alpha = 3$
ALO	—
ODE1	$F = 0.5$, Cr = 0.9, JR = 0.3

4.2. Experimental Test 2: Multimodal Functions. Different from unimodal function, multimodal test functions have multiple local optimal solutions and thus are commonly adopted by researchers for testing the exploration ability of an algorithm. Tables 7 and 8 provide the recorded results of statistical analysis over 10 independent runs for $n = 30$ and 100, respectively. From these tables, it is clear that ELAPO can get better level of accuracy for most of test functions compared with other six algorithms. Particularly, ELAPO is able to obtain the exact true values of F17, F18, F21, F22, and F24. It is also interesting to find that ELAPO is still better than LAPO on F20 although both cannot match with SSA at all dimensions involved. Similar to the observation in the unimodal functions, ELAPO seems to be insensitive to the increase of dimensional number.

The convergence performance of all algorithms for several multimodal benchmark functions at $n = 30$ and 100 are presented in Figures 3 and 4, respectively. As can be found in these figures, ELAPO always has the fastest convergence rate and can reach the best (at least comparable) convergence precision in comparison to other six algorithms. For some multimodal functions such as F13, F14, and F16, the convergence performance of LAPO is unsatisfactory, while the global convergence ability of ELAPO is improved greatly. This is mainly contributed by the quasi-opposition-based strategy in which new opposite test points are generated according to a portion of randomly selected test points and both are simultaneously employed for global searching.

4.3. Experimental Test 3: CEC 2014 Benchmark Functions. In this experimental study, most intensely investigated benchmark functions used in IEEE CEC 2014 are considered for evaluating both exploration and exploitation capabilities of ELAPO. Seven CEC 2014 functions are considered, which consist of several novel basic problems (e.g., with shifting and rotation) and hybrid and composite test problems. These modern benchmark functions are specially developed with complex features; consequently, all the algorithms can hardly reach the global optimum. However, as per statistical results obtained from different algorithms through 10 independent runs in Tables 9 and 10, ELAPO is able to yield highly competitive results for all CEC 2014 functions under consideration as compared with other six algorithms. For example, the mean error of F27 is as low as a level of $10E-1$

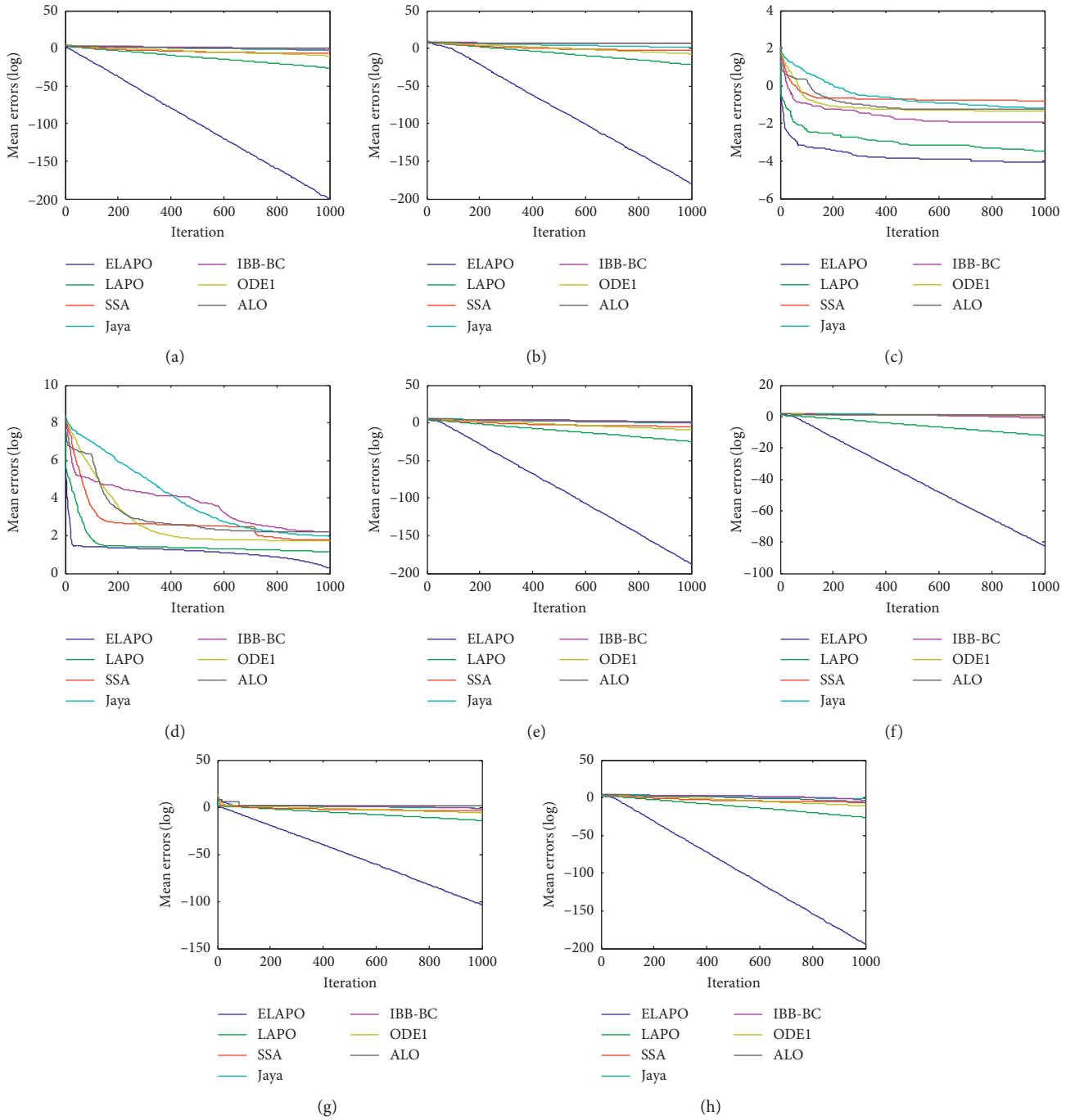

Figure 1: Average convergence curves for the selected unimodal functions ($n = 30$). (a) F1. (b) F4. (c) F5. (d) F6. (e) F7. (f) F8. (g) F9. (h) F10.

for ELAPO, while the corresponding mean errors are dramatically larger for LAPO ($10E+2$), ODE1 ($10E+3$), SSA, IBB-BC and ALO ($10E+4$), and Jaya ($10E+9$). As the number of dimension increases, all algorithms produce relatively larger mean errors for F26–29 with higher standard deviations, among which ELAPO still ranks No. 1. For the composite functions (F30–32), the increase of dimension seems to not affect the statistical results of all algorithms and ELAPO tends to give slightly better results than other six algorithms.

Figures 5 and 6 show the average convergence curves for four selected CEC 2014 benchmark functions at $n = 30$ and 100, respectively. It is clear that ELAPO has promising convergence behavior compared with other six algorithms, and thus ELAPO proves to be the best among all algorithms on seven CEC 2014 functions. This serves as a further confirmation that ELAPO possesses excellent balance between exploration and exploitation.

4.4. Effectiveness of the Two Strategies. In order to verify the effectiveness of the two strategies in the proposed algorithm, this subsection performs the previous three experiments for

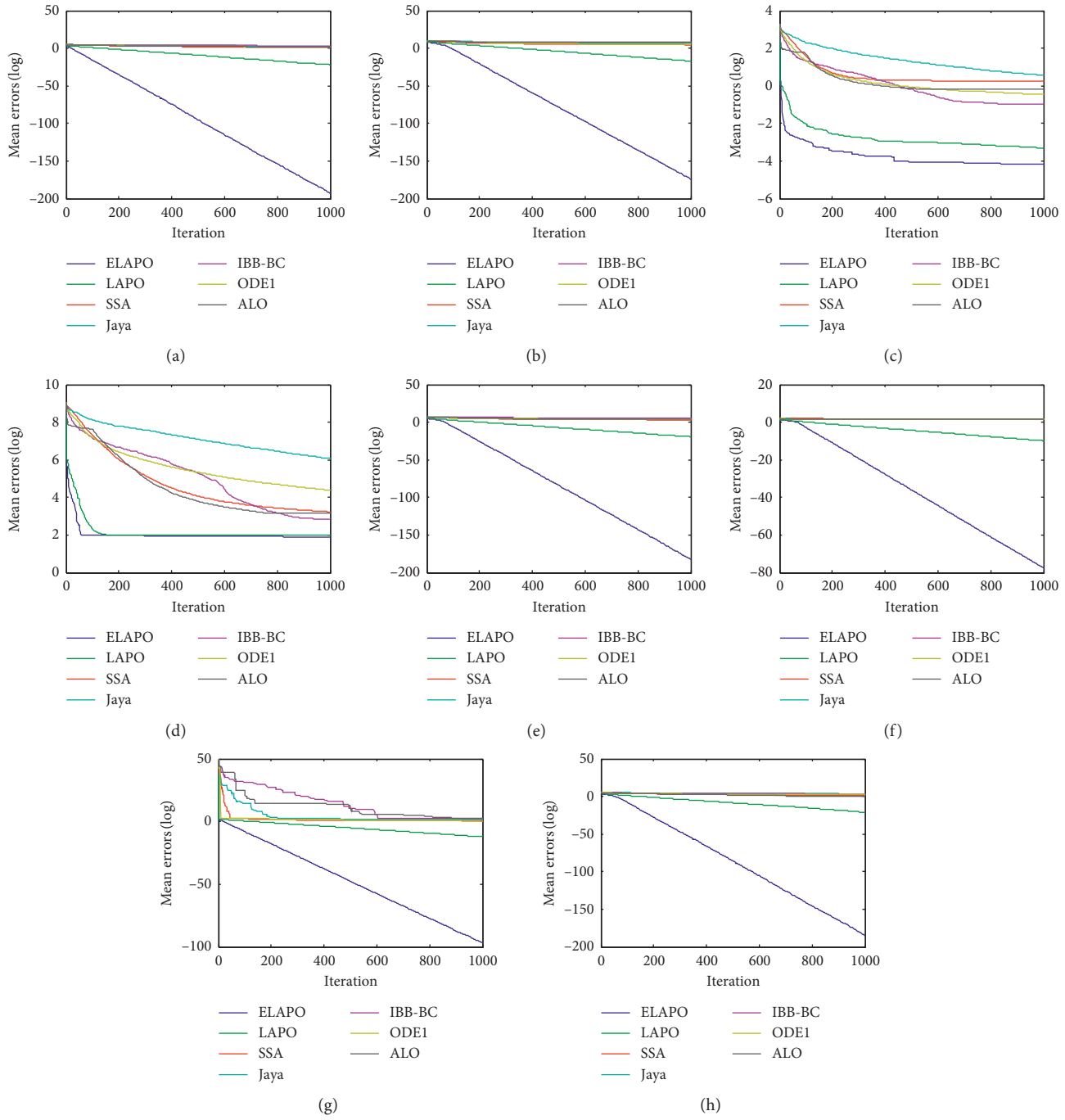

Figure 2: Average convergence curves for the selected unimodal functions ($n = 100$). (a) F1. (b) F4. (c) F5. (d) F6. (e) F7. (f) F8. (g) F9. (h) F10.

ELAPO, LAPO with quasi-opposition-based learning only (denoted as ELAPO1), and LAPO with dimensional search strategy only (denoted as ELAPO2), respectively. The statistical results (the minimum, mean and maximum fitness values, and the standard deviation) of different ELAPO variants are recorded in Tables 11–16 for various test functions at $n = 30$ and 100. For each function, the overall best results among the three algorithms are highlighted in bold.

For most of the unimodal functions, as shown in Tables 11 and 12, ELAPO outperforms the other two variants in terms of the minimum, mean and maximum fitness values, and the standard deviation. This confirms that for most of the functions, both strategies take effects on enhancing the global search ability and the contribution of the quasi-opposition-based learning strategy is more important. As for F6, it seems that the dimensional search strategy has bigger contributions on the exploitation ability of ELAPO. It is also noted that ELAPO and its two variants have almost the same statistical results because, as per Table 5, the basic LAPO has already converged to desirable accuracy and thus the two strategies seem to have no effects.

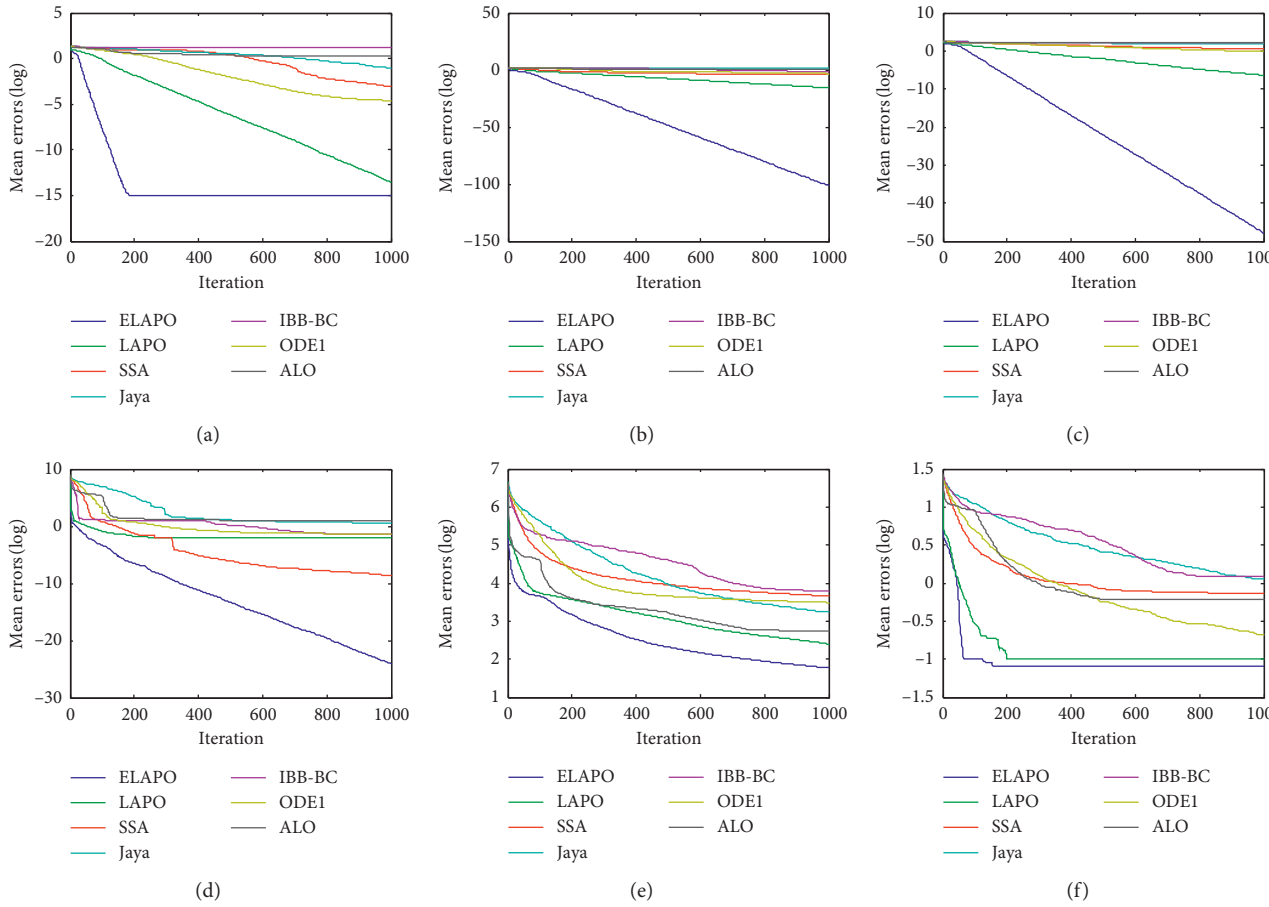

Figure 3: Average convergence curves for the selected multimodal functions ($n = 30$). (a) F12. (b) F13. (c) F14. (d) F16. (e) F19. (f) F23.

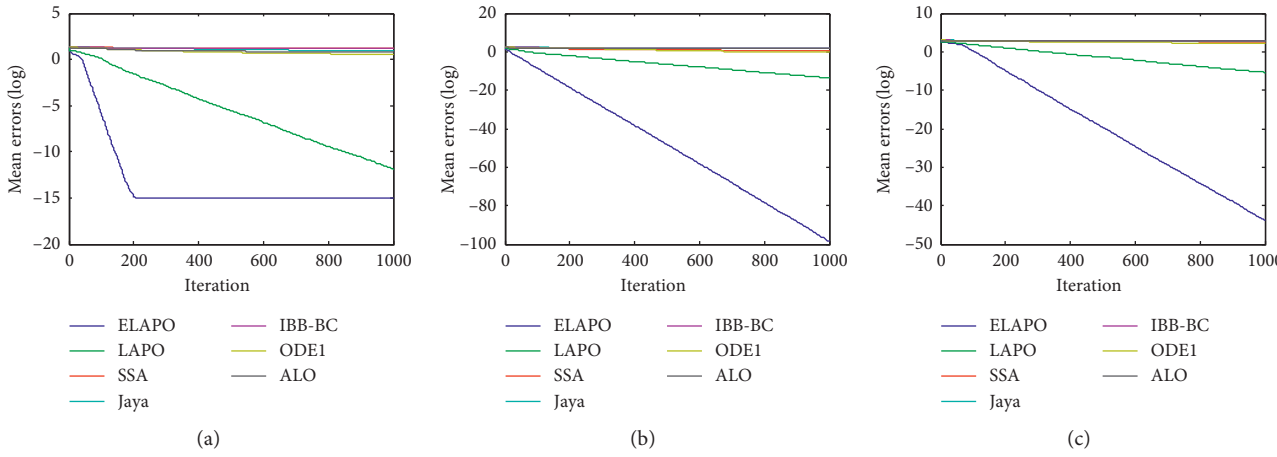

Figure 4: Continued.

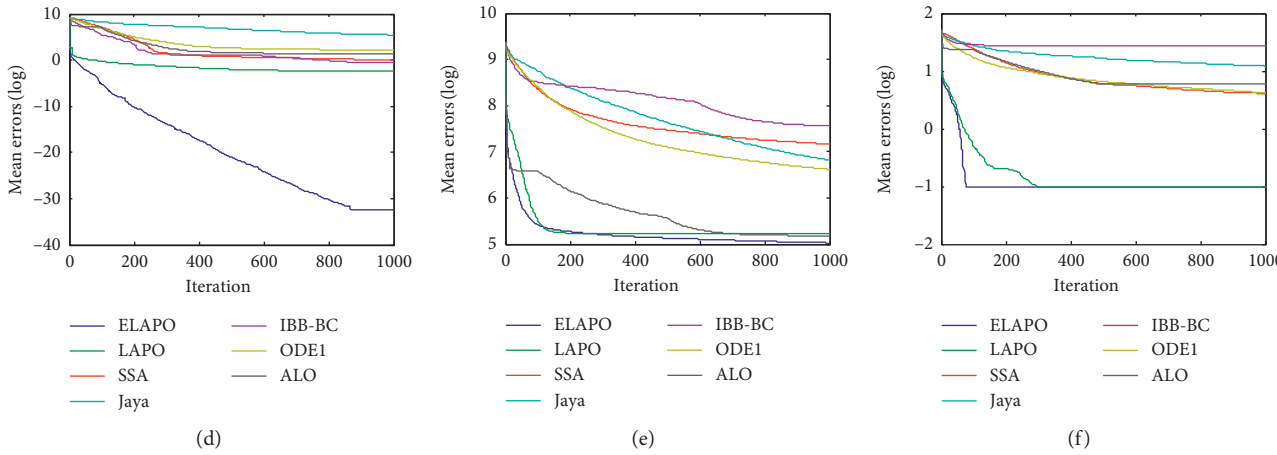

Figure 4: Average convergence curves for the selected multimodal functions ($n = 100$). (a) F12. (b) F13. (c) F14. (d) F16. (e) F19. (f) F23.

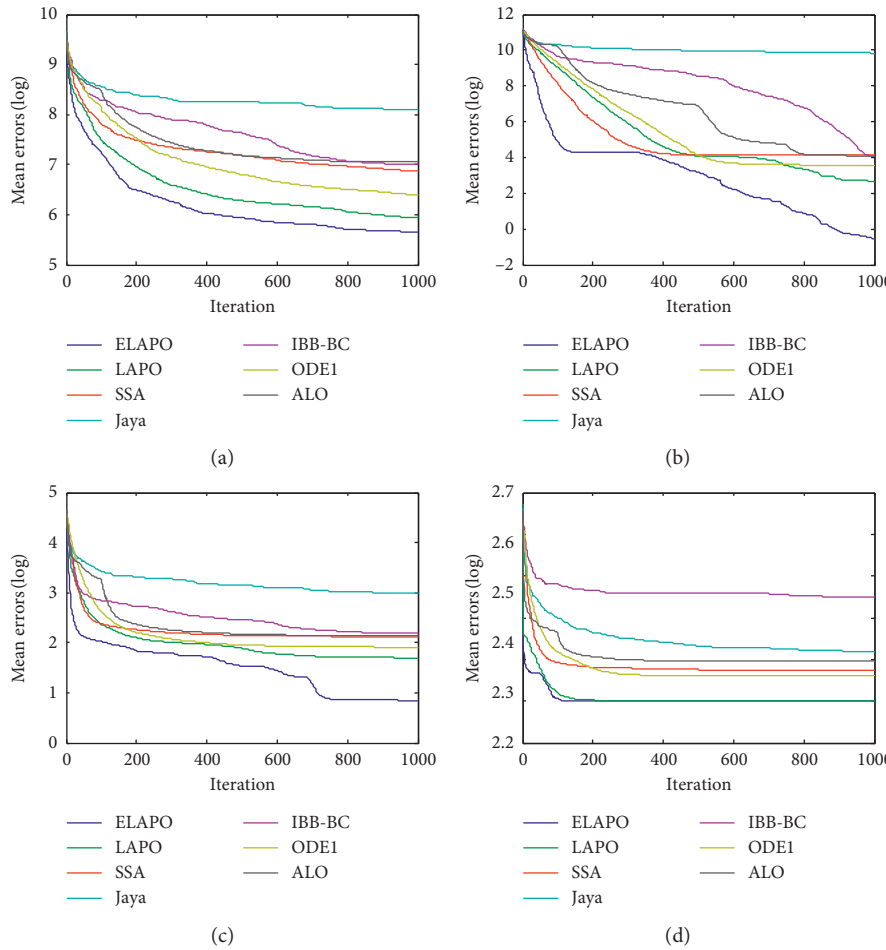

Figure 5: Average convergence curves for the selected CEC 2014 functions ($n = 30$). (a) F26. (b) F27. (c) F28. (d) F31.

Tables 13 and 14 show the minimum, mean and maximum fitness values, and the standard deviation of the multimodal benchmark functions using ELAPO and its two variants. It is clear to find that for most of benchmark functions, both strategies are beneficial to the global search performance and their individual contributions vary for a specific function. For example, the quasi-opposition-based learning strategy has more important effects on F13, F14, and F17; the dimensional search strategy plays a more important role on F15, F16, and F20; both strategies have almost equal contributions on F12 and F23. For F21, F22, and F24, both strategies seem to

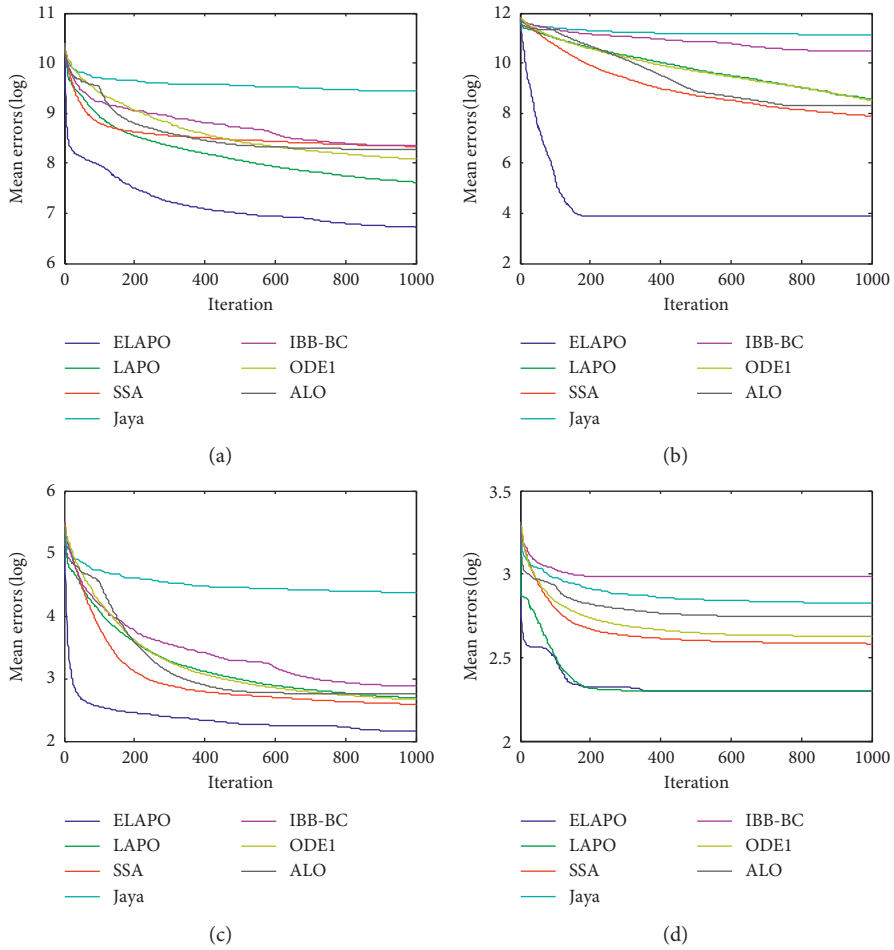

Figure 6: Average convergence curves for the selected CEC 2014 functions ($n = 100$). (a)F26. (b) F27. (c) F28. (d) F31.

Table 5: Statistical results obtained by different algorithms through 10 independent runs for unimodal benchmark functions at $n = 30$.

Function		ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F1	Mean	**7.5637E − 200**	5.6012$E-27$	1.1428$E-07$	2.2752$E-03$	1.3687$E-02$	1.0558$E-10$	4.4514$E-01$
	Std	**0.0000E + 00**	8.4361$E-27$	4.8364$E-08$	8.0246$E-04$	1.5007$E-02$	3.2551$E-10$	7.8851$E-01$
F2	Mean	**6.6667E − 01**	**6.6667E − 01**	9.4666$E-01$	1.5883$E+00$	1.9627$E+00$	8.2956$E-01$	1.9322$E+00$
	Std	**1.1703E − 16**	**2.6825E − 15**	9.0549$E-01$	1.4893$E+00$	3.0291$E+00$	3.4162$E-01$	1.6808$E+00$
F3	Mean	**0.0000E + 00**	**0.0000E + 00**	4.8854$E-11$	1.0043$E-06$	9.5870$E-01$	0.0000$E+00$	1.1247$E-11$
	Std	**0.0000E + 00**	**0.0000E + 00**	2.4693$E-11$	3.2914$E-07$	1.6557$E-02$	6.3238$E-16$	5.3013$E-12$
F4	Mean	**1.8129E − 180**	3.6621$E-22$	1.4830$E-03$	1.5201$E+01$	3.1383$E+06$	4.6192$E-08$	2.4627$E+06$
	Std	**0.0000E + 00**	9.1437$E-22$	1.3430$E-03$	5.5752$E+00$	1.5412$E+06$	9.0067$E-08$	1.2642$E+06$
F5	Mean	**8.6010E − 05**	3.3511$E-04$	1.4931$E-01$	6.6240$E-02$	1.1173$E-02$	4.5462$E-02$	5.6409$E-02$
	Std	**8.9264E − 05**	2.3261$E-04$	4.1331$E-02$	8.9128$E-03$	1.7186$E-03$	1.1837$E-02$	2.0886$E-02$
F6	Mean	**1.8767E + 00**	1.3696$E+01$	6.0642$E+01$	9.2536$E+01$	1.6858$E+02$	5.2937$E+01$	1.6217$E+02$
	Std	**8.1532E − 01**	9.9927$E-01$	3.5585$E+01$	4.5618$E+01$	2.2938$E+02$	2.9763$E+01$	1.6493$E+02$
F7	Mean	**2.5280E − 188**	1.3524$E-25$	9.7179$E-06$	2.7516$E-01$	7.1328$E-01$	5.3976$E-10$	1.7832$E+01$
	Std	**0.0000E + 00**	1.3883$E-25$	5.6762$E-06$	1.1147$E-01$	9.4554$E-01$	1.2392$E-09$	2.0860$E+01$
F8	Mean	**5.2137E − 83**	8.3427$E-13$	2.6260$E+00$	6.9426$E+00$	1.6891$E-01$	7.1509$E+00$	8.1078$E+00$
	Std	**8.6122E − 83**	5.5630$E-13$	6.1185$E-01$	3.4493$E+00$	1.3435$E-01$	3.2087$E+00$	2.2939$E+00$
F9	Mean	**3.6582E − 104**	6.5210$E-15$	1.9378$E-04$	7.0569$E-02$	5.0370$E-02$	7.7739$E-07$	3.4909$E+01$
	Std	**5.3056E − 104**	1.4104$E-14$	5.6284$E-05$	2.2004$E-02$	2.9201$E-02$	3.0011$E-07$	4.8480$E+01$
F10	Mean	**8.1828E − 195**	5.8616$E-27$	1.0567$E-06$	1.9651$E-02$	3.5197$E-04$	1.5345$E-11$	6.6424$E-07$
	Std	**0.0000E + 00**	8.6367$E-27$	8.7349$E-07$	6.2693$E-03$	1.4587$E-04$	1.5332$E-11$	5.6633$E-07$
F11	Mean	**0.0000E + 00**	1.4564$E-103$	1.3567$E-41$	3.4416$E-13$	6.3934$E-08$	9.7599$E-13$	1.3755$E-07$
	Std	**0.0000E + 00**	3.7807$E-103$	3.3385$E-41$	7.5948$E-13$	6.9210$E-08$	2.3650$E-12$	9.0853$E-08$

Table 6: Statistical results obtained by different algorithms through 10 independent runs for unimodal benchmark functions at $n = 100$.

Function		ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F1	Mean	**5.2007E − 193**	1.3425$E-22$	1.6628$E+01$	6.9384$E+02$	1.8003$E+03$	1.8499$E+01$	1.1049$E+02$
	Std	**0.0000E + 00**	2.7548$E-22$	3.1328$E+00$	1.5143$E+02$	1.1134$E+03$	1.2793$E+01$	3.4927$E+01$
F2	Mean	**6.6667E − 01**	**6.6667E − 01**	9.3976$E+01$	1.7780$E+04$	5.8507$E+01$	2.4492$E+02$	6.5552$E+01$
	Std	**1.1703E − 16**	**3.1465E − 07**	1.8308$E+01$	9.2943$E+03$	5.5585$E+01$	1.0120$E+02$	1.6325$E+01$
F3	Mean	**0.0000E + 00**	**0.0000E + 00**	2.1775$E-03$	8.7150$E-02$	1.0000$E+00$	3.7746$E-03$	1.7589$E-06$
	Std	**0.0000E + 00**	**0.0000E + 00**	4.0246$E-04$	1.9928$E-02$	1.4447$E-06$	4.3774$E-03$	7.2246$E-07$
F4	Mean	**3.5068E − 174**	7.6209$E-18$	5.7986$E+04$	3.3097$E+06$	6.4465$E+07$	1.1275$E+05$	3.7806$E+07$
	Std	**0.0000E + 00**	1.4480$E-17$	1.0700$E+04$	1.8257$E+06$	1.8345$E+07$	1.3526$E+05$	1.0211$E+07$
F5	Mean	**6.9030E − 05**	5.0481$E-04$	1.7809$E+00$	3.7596$E+00$	1.1013$E-01$	3.4775$E-01$	6.9402$E-01$
	Std	**1.2101E − 04**	3.4086$E-04$	2.8494$E-01$	1.2333$E+00$	3.0997$E-02$	1.6561$E-01$	8.1848$E-02$
F6	Mean	**7.8513E + 01**	9.4177$E+01$	1.7444$E+03$	1.1493$E+06$	7.2389$E+02$	2.3396$E+04$	1.4132$E+03$
	Std	**1.6552E + 00**	8.9340$E-01$	5.2182$E+02$	4.4284$E+05$	4.8630$E+02$	2.7369$E+04$	1.2610$E+03$
F7	Mean	**1.1968E − 182**	6.3973$E-20$	1.8395$E+03$	6.3193$E+04$	1.3746$E+05$	2.9564$E+03$	1.2062$E+04$
	Std	**0.0000E + 00**	1.2555$E-19$	3.3413$E+02$	1.2607$E+04$	7.9074$E+04$	2.1890$E+03$	3.5187$E+03$
F8	Mean	**4.0687E − 78**	1.5876$E-10$	5.6812$E+01$	5.0409$E+01$	5.3766$E+01$	2.4541$E+01$	2.7036$E+01$
	Std	**3.8556E − 78**	8.3377$E-11$	3.3342$E+00$	4.5067$E+00$	4.7994$E+00$	4.2786$E+00$	3.7373$E+00$
F9	Mean	**1.7153E − 97**	6.0151$E-13$	3.7898$E+00$	4.0015$E+01$	2.2845$E+02$	2.5136$E+00$	2.5022$E+02$
	Std	**2.6804E − 97**	7.2192$E-13$	3.3142$E-01$	9.5389$E+00$	2.4337$E+01$	1.1042$E+00$	1.8829$E+02$
F10	Mean	**2.1712E − 185**	1.0305$E-21$	4.5309$E+01$	2.0729$E+03$	1.5794$E+03$	1.5935$E+02$	7.2968$E-01$
	Std	**0.0000E + 00**	1.3847$E-21$	8.5410$E+00$	5.9716$E+02$	1.4768$E+03$	2.6888$E+02$	3.5403$E-01$
F11	Mean	**0.0000E + 00**	6.1091$E-103$	4.7120$E-29$	2.4029$E-07$	8.6095$E-08$	1.6291$E-10$	1.5573$E-07$
	Std	**0.0000E + 00**	1.6435$E-102$	1.1490$E-28$	3.3712$E-07$	5.7812$E-08$	5.1012$E-10$	1.0685$E-07$

Table 7: Statistical results obtained by different algorithms through 10 independent runs for multimodal benchmark functions at $n = 30$.

Fun		ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F12	Mean	**8.8818E − 16**	2.8955$E-14$	9.2571$E-04$	9.1487$E-02$	1.7601$E+01$	2.5302$E-05$	1.6649$E+00$
	Std	**0.0000E + 00**	2.4640$E-14$	9.2067$E-04$	7.7490$E-02$	6.9501$E-01$	7.7013$E-05$	7.7554$E-01$
F13	Mean	**6.1910E − 102**	2.2608$E-16$	1.1359$E-04$	1.8262$E+01$	5.4072$E-02$	5.6933$E-04$	5.2074$E+00$
	Std	**1.7386E − 101**	1.6391$E-16$	5.9272$E-05$	8.6301$E+00$	4.1391$E-02$	1.2644$E-03$	3.1551$E+00$
F14	Mean	**1.4029E − 48**	4.0494$E-07$	3.4234$E+00$	5.1721$E+01$	1.1805$E+02$	6.8554$E-01$	1.3231$E+02$
	Std	**1.7605E − 48**	2.6617$E-07$	9.0812$E-01$	9.8384$E+00$	1.7959$E+01$	1.7645$E-01$	2.0228$E+01$
F15	Mean	1.4563$E+00$	9.4103$E+00$	**9.0346E − 01**	1.2447$E+01$	1.2934$E+01$	1.2364$E+01$	1.2030$E+01$
	Std	4.3965$E-01$	5.3815$E-01$	**3.2252E − 01**	2.9666$E-01$	5.3194$E-01$	2.5704$E-01$	7.5729$E-01$
F16	Mean	**8.4188E − 25**	1.0367$E-02$	2.2266$E-09$	4.1310$E+00$	4.1888$E-02$	4.2953$E-02$	9.0097$E+00$
	Std	**1.5385E − 24**	3.2783$E-02$	1.5417$E-09$	1.6123$E+00$	5.4071$E-02$	5.2502$E-02$	3.2448$E+00$
F17	Mean	**0.0000E + 00**	7.3960$E-04$	2.2384$E-02$	2.9603$E-01$	5.1123$E-02$	5.4206$E-03$	1.0343$E-02$
	Std	**0.0000E + 00**	2.3388$E-03$	1.3974$E-02$	2.2107$E-01$	6.7979$E-02$	6.1229$E-03$	1.0090$E-02$
F18	Mean	**0.0000E + 00**	1.6903$E+01$	5.7452$E+00$	2.1310$E+01$	2.5789$E+01$	2.0659$E+01$	1.8115$E+01$
	Std	**0.0000E + 00**	6.2598$E+00$	1.6003$E+00$	8.2816$E-01$	4.4569$E-01$	8.8290$E-01$	2.5728$E+00$
F19	Mean	**5.8977E + 01**	2.5151$E+02$	4.5305$E+03$	1.7584$E+03$	6.3780$E+03$	3.0747$E+03$	5.5289$E+02$
	Std	**2.4102E + 01**	1.0785$E+02$	3.6141$E+03$	2.6629$E+03$	6.1551$E+03$	1.1640$E+03$	1.7077$E+02$
F20	Mean	2.0743$E-03$	1.5861$E-02$	**3.1866E − 04**	7.3750$E-02$	3.3350$E-03$	3.9377$E-03$	8.3297$E-03$
	Std	2.6236$E-03$	1.5529$E-02$	**3.6113E − 04**	2.9955$E-02$	1.3663$E-03$	2.3799$E-03$	4.6551$E-03$
F21	Mean	**0.0000E + 00**	**0.0000E + 00**	8.1814$E-06$	2.2150$E+02$	5.3244$E+01$	1.5228$E+02$	7.8900$E+01$
	Std	**0.0000E + 00**	**0.0000E + 00**	1.2639$E-05$	2.0149$E+01$	1.7451$E+01$	2.5607$E+01$	2.5535$E+01$
F22	Mean	**0.0000E + 00**	1.2264$E+01$	1.6897$E-01$	2.0120$E+02$	4.6472$E+01$	1.1726$E+02$	8.5910$E+01$
	Std	**0.0000E + 00**	2.5856$E+01$	2.7200$E-01$	1.4093$E+01$	1.2995$E+01$	4.0305$E+01$	5.0467$E+01$
F23	Mean	**7.9899E − 02**	9.9873$E-02$	7.1987$E-01$	1.1289$E+00$	1.2306$E+00$	2.1167$E-01$	6.0987$E-01$
	Std	**4.2110E − 02**	9.2544$E-09$	6.3246$E-02$	7.3969$E-02$	2.7575$E-01$	3.1132$E-02$	7.3786$E-02$
F24	Mean	**0.0000E + 00**	**0.0000E + 00**	2.0537$E-02$	1.6488$E+00$	1.4966$E+01$	1.8835$E-02$	1.7219$E+01$
	Std	**0.0000E + 00**	**0.0000E + 00**	8.4213$E-03$	7.2301$E-01$	4.3592$E+00$	3.8947$E-02$	3.2418$E+00$
F25	Mean	**3.4556E + 02**	6.2050$E+02$	3.6018$E+02$	8.1492$E+02$	8.1624$E+02$	7.6337$E+02$	7.3879$E+02$
	Std	**3.3722E + 01**	1.4539$E+02$	1.1172$E+02$	1.9261$E+01$	2.0303$E+01$	5.3897$E+01$	3.2218$E+01$

Table 8: Statistical results obtained by different algorithms through 10 independent runs for multimodal benchmark functions at $n = 100$.

Fun		ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F12	Mean	**8.8818E − 16**	1.2518E − 12	1.6404E + 01	7.5892E + 00	1.8804E + 01	3.7259E + 00	6.8021E + 00
	Std	**0.0000E + 00**	1.9773E − 12	7.3335E + 00	6.3212E − 01	2.1140E − 01	9.9069E − 01	1.3361E + 00
F13	Mean	**3.6083E − 99**	2.2834E − 14	3.9731E + 00	5.3830E + 01	5.3426E + 01	4.6040E − 01	3.6655E + 01
	Std	**5.4747E − 99**	1.5184E − 14	1.0066E + 00	8.5028E + 00	8.0583E + 00	3.3718E − 01	1.1067E + 01
F14	Mean	**1.7929E − 44**	4.5968E − 06	4.1324E + 02	6.3538E + 02	7.7868E + 02	1.4752E + 02	6.0052E + 02
	Std	**2.8722E − 44**	2.6509E − 06	5.0767E + 01	4.8142E + 01	3.4226E + 01	2.1587E + 01	5.0591E + 01
F15	Mean	**7.2183E + 00**	4.0465E + 01	2.7036E + 01	4.6481E + 01	4.4956E + 01	4.5921E + 01	4.1953E + 01
	Std	**8.8821E + 00**	9.3027E − 01	1.9176E + 00	3.0431E − 01	8.6263E − 01	1.0532E + 00	2.1483E + 00
F16	Mean	**4.7116E − 33**	3.3016E − 03	9.3548E − 01	2.7447E + 05	3.2179E − 01	1.5236E + 02	2.2466E + 01
	Std	**1.9082E − 48**	9.8103E − 03	3.5109E − 01	4.6952E + 05	2.2593E − 01	2.5273E + 02	7.5516E + 00
F17	Mean	**0.0000E + 00**	**0.0000E + 00**	4.9065E − 01	1.4511E + 00	2.2854E + 00	3.9525E − 01	1.7248E − 01
	Std	**0.0000E + 00**	**0.0000E + 00**	6.1173E − 02	9.6738E − 02	3.8727E − 01	1.6189E − 01	3.8321E − 02
F18	Mean	**2.4937E + 01**	8.3464E + 01	2.9867E + 01	8.6792E + 01	9.2907E + 01	8.6137E + 01	6.7602E + 01
	Std	**3.2165E + 01**	3.3841E + 00	2.0862E + 00	1.0848E + 00	1.0617E + 00	9.8553E − 01	6.6475E + 00
F19	Mean	**1.0585E + 05**	1.6900E + 05	1.4604E + 07	6.4959E + 06	3.7560E + 07	4.1557E + 06	1.5224E + 05
	Std	**4.1853E + 04**	2.9643E + 02	4.0214E + 06	1.5787E + 06	1.4721E + 07	1.1991E + 06	1.1620E + 04
F20	Mean	7.3630E − 02	1.5070E + 00	**4.4115E − 02**	2.8379E + 00	2.9689E − 01	1.7898E − 01	3.3727E − 01
	Std	1.5569E − 01	3.5022E − 01	**1.4144E − 02**	4.1322E − 01	1.3280E − 01	5.3857E − 02	9.9795E − 02
F21	Mean	**0.0000E + 00**	**0.0000E + 00**	1.3590E + 02	9.3314E + 02	4.9346E + 02	7.0721E + 02	2.4664E + 02
	Std	**0.0000E + 00**	**0.0000E + 00**	1.1534E + 01	5.9504E + 01	4.5275E + 01	8.4180E + 01	5.1974E + 01
F22	Mean	**0.0000E + 00**	**0.0000E + 00**	9.0892E + 01	9.5631E + 02	5.4737E + 02	7.5311E + 02	4.1579E + 02
	Std	**0.0000E + 00**	**0.0000E + 00**	1.3962E + 01	6.2407E + 01	5.9740E + 01	4.7769E + 01	1.0978E + 02
F23	Mean	**9.9873E − 02**	**9.9873E − 02**	4.0825E + 00	1.2141E + 01	2.7545E + 01	4.1911E + 00	5.8799E + 00
	Std	**7.7820E − 17**	**9.5023E − 09**	4.1814E − 01	8.9201E − 01	1.7224E + 00	5.9174E − 01	8.4301E − 01
F24	Mean	**0.0000E + 00**	**0.0000E + 00**	1.5556E + 01	5.0470E + 01	1.2573E + 02	1.0650E + 01	8.9877E + 01
	Std	**0.0000E + 00**	**0.0000E + 00**	2.0040E + 00	3.8831E + 00	6.1093E + 00	1.7570E + 00	6.6157E + 00
F25	Mean	**3.6192E + 03**	4.5995E + 03	1.0766E + 04	3.5395E + 06	1.0381E + 04	6.5557E + 04	1.2502E + 04
	Std	**8.1059E + 02**	9.5869E − 13	2.6120E + 02	1.8511E + 06	2.2635E + 02	6.1928E + 04	6.4827E + 02

Table 9: Statistical results obtained by different algorithms through 10 independent runs for CEC 2014 benchmark functions at $n = 30$.

Fun		ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F26	Mean	**4.4625E + 05**	8.6316E + 05	7.4184E + 06	1.2764E + 08	1.0313E + 07	2.4740E + 06	1.1687E + 07
	Std	**4.1890E + 05**	4.7927E + 05	4.2619E + 06	4.7324E + 07	7.9033E + 06	2.0985E + 06	4.7916E + 06
F27	Mean	**3.0019E − 01**	4.8777E + 02	1.2834E + 04	6.8890E + 09	1.2215E + 04	3.6652E + 03	1.2669E + 04
	Std	**5.2289E − 01**	4.4184E + 02	1.1687E + 04	8.9349E + 08	6.3267E + 03	5.1977E + 03	7.5825E + 03
F28	Mean	**7.0443E + 00**	4.9654E + 01	1.2784E + 02	9.7486E + 02	1.5431E + 02	7.9861E + 01	1.3876E + 02
	Std	**2.1148E + 01**	3.3024E + 01	3.9603E + 01	1.8138E + 02	1.0217E + 01	1.9594 E + 01	4.1850E + 01
F29	Mean	1.6076E + 05	2.9803E + 05	1.2743E + 06	6.8526E + 06	6.1188E + 05	**2.4020E + 04**	1.1616E + 06
	Std	1.0464E + 05	1.4700E + 05	6.7634E + 05	3.2198E + 06	4.6454E + 05	**2.4324E + 04**	7.6552E + 05
F30	Mean	**3.1524E + 02**	**3.1524E + 02**	3.1526E + 02	3.6242E + 02	4.0373E + 02	3.1525E + 02	3.2299E + 02
	Std	**4.7935E − 13**	**9.3896E − 12**	1.5792E − 02	1.1067E + 01	3.7105E + 01	3.7343E − 03	3.8656E + 00
F31	Mean	**2.0000E + 02**	2.0001E + 02	2.3645E + 02	2.6188E + 02	3.5442E + 02	2.2897E + 02	2.4859E + 02
	Std	**3.5666E − 04**	1.7300E − 03	8.9457E + 00	8.8881E + 00	1.5967E + 01	5.5403E + 00	5.1698E + 00
F32	Mean	**2.0000E + 02**	**2.0000E + 02**	2.0932E + 02	2.2711E + 02	2.4501E + 02	2.0312E + 02	2.2389E + 02
	Std	**0.0000E + 00**	**2.1437E − 13**	5.7620E + 00	3.4992E + 00	1.1029E + 01	3.2954E − 01	4.8761E + 00

have no effect. This is because, as per Table 8, the basic LAPO has already obtained the exact optimum. It is also interesting to find that negative effect may be exerted on the global optimization capability. Taking F25 as an example, the best minimum, mean, and maximum fitness values all lied on ELAPO2. In other words, the dimensional search strategy and the quasi-opposition-based learning strategy take positive and negative effects on F25, respectively; the combination of both (ELAPO) can only achieve a middle place of statistical results between

Table 10: Statistical results obtained by different algorithms through 10 independent runs for CEC 2014 benchmark functions at $n = 100$.

Fun		ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F26	Mean	**5.2160E + 06**	4.1519E + 07	2.1696E + 08	2.7847E + 09	2.2334E + 08	1.1944E + 08	1.9248E + 08
	Std	**3.1061E + 06**	1.0226E + 07	6.2278E + 07	5.3126E + 08	4.9251E + 07	4.6916E + 07	5.2322E + 07
F27	Mean	**7.6528E + 03**	3.6048E + 08	7.7122E + 07	1.3095E + 11	2.9545E + 10	3.1169E + 08	1.9225E + 08
	Std	**4.6668E + 03**	4.0022E + 08	1.5673E + 07	1.4234E + 10	1.6259E + 10	2.7877E + 08	1.2626E + 08
F28	Mean	**1.4633E + 02**	4.8679E + 02	3.8588E + 02	2.3145E + 04	7.7160E + 02	4.5681E + 02	5.5644E + 02
	Std	**4.9530E + 01**	8.2735E + 01	4.7462E + 01	4.8763E + 03	1.1464E + 02	6.2699E + 01	5.1636E + 01
F29	Mean	**2.0346E + 06**	3.8720E + 06	2.3420E + 07	2.7397E + 08	9.2325E + 06	9.7017E + 06	8.6593E + 06
	Std	**1.2767E + 06**	2.1761E + 06	1.3998E + 07	3.4244E + 07	4.6827E + 06	2.9845E + 06	3.3179E + 06
F30	Mean	**3.4508E + 02**	3.5079E + 02	3.5220E + 02	1.1047E + 03	9.8448E + 02	3.5205E + 02	4.8240E + 02
	Std	**6.2281E + 01**	1.2145E + 00	2.1181E + 00	1.5451E + 02	1.4673E + 02	3.0127E + 00	2.5739E + 01
F31	Mean	**2.0000E + 02**	2.0002E + 02	3.8192E + 02	6.6879E + 02	9.6557E + 02	4.2456E + 02	5.5390E + 02
	Std	**1.0968E − 03**	5.9258E − 03	3.3080E + 00	3.1583E + 01	3.6526E + 01	5.7272E + 00	4.5437E + 01
F32	Mean	**2.0000E + 02**	**2.0000E + 02**	2.7324E + 02	5.6117E + 02	5.2303E + 02	2.6458E + 02	3.3362E + 02
	Std	**0.0000E + 00**	**1.4101E − 11**	1.9475E + 01	4.0497E + 01	5.1939E + 01	8.5014E + 00	1.5205E + 01

Table 11: Statistical results for unimodal benchmark functions of different ELAPO ($n = 30$).

Function	Algorithm	Min.	Mean	Max.	Std
F1	ELAPO	**4.3947E − 207**	**7.5637E − 200**	**7.5041E − 199**	**0.0000E + 00**
	ELAPO1	1.0983E − 196	2.0585E − 192	1.1337E − 191	**0.0000E + 00**
	ELAPO2	1.3264E − 45	9.8461E − 45	4.1225E − 44	1.2223E − 44
F2	ELAPO	**6.6667E − 01**	**6.6667E − 01**	**6.6667E − 01**	1.1703E − 16
	ELAPO1	**6.6667E − 01**	**6.6667E − 01**	**6.6667E − 01**	**9.7912E − 17**
	ELAPO2	**6.6667E − 01**	**6.6667E − 01**	**6.6667E − 01**	5.5299E − 14
F3	ELAPO	**−1.0000E + 00**	**−1.0000E + 00**	**−1.0000E + 00**	**0.0000E + 00**
	ELAPO1	**−1.0000E + 00**	**−1.0000E + 00**	**−1.0000E + 00**	**0.0000E + 00**
	ELAPO2	**−1.0000E + 00**	**−1.0000E + 00**	**−1.0000E + 00**	**0.0000E + 00**
F4	ELAPO	**1.0516E − 186**	**1.8129E − 180**	**1.0021E − 179**	**0.0000E + 00**
	ELAPO1	3.4388E − 184	1.3329E − 178	6.4548E − 178	**0.0000E + 00**
	ELAPO2	1.5771E − 42	1.5959E − 29	1.5959E − 28	5.0466E − 29
F5	ELAPO	**1.4582E − 05**	**8.6010E − 05**	**2.4099E − 04**	**8.9264E − 05**
	ELAPO1	2.9369E − 05	1.8099E − 04	5.7794E − 04	1.6862E − 04
	ELAPO2	1.1022E − 04	2.5442E − 04	6.2774E − 04	1.5210E − 04
F6	ELAPO	8.4106E − 01	**1.8767E + 00**	**3.5521E + 00**	**8.1532E − 01**
	ELAPO1	9.7267E + 00	1.2179E + 01	1.3300E + 01	1.0108E + 00
	ELAPO2	**3.4946E − 02**	2.7249E + 00	4.7309E + 00	1.5030E + 00
F7	ELAPO	**3.3224E − 193**	**2.5280E − 188**	**1.9516E − 187**	**0.0000E + 00**
	ELAPO1	1.8360E − 187	1.6201E − 184	9.7195E − 184	**0.0000E + 00**
	ELAPO2	2.7107E − 42	3.5402E − 41	1.3260E − 40	3.8879E − 41
F8	ELAPO	**2.1127E − 84**	**5.2137E − 83**	**2.6909E − 82**	**8.6122E − 83**
	ELAPO1	1.1547E − 83	2.3096E − 81	1.0418E − 80	3.6979E − 81
	ELAPO2	2.2641E − 15	4.7375E − 15	7.9828E − 15	2.0046E − 15
F9	ELAPO	**1.1064E − 106**	**3.6582E − 104**	**1.8075E − 103**	**5.3056E − 104**
	ELAPO1	6.8578E − 100	1.1707E − 98	4.2746E − 98	1.3897E − 98
	ELAPO2	1.8562E − 23	4.1160E − 23	1.2816E − 22	3.2696E − 23
F10	ELAPO	**5.7179E − 200**	**8.1828E − 195**	**5.1517E − 194**	**0.0000E + 00**
	ELAPO1	1.4263E − 190	9.8501E − 186	9.8128E − 185	**0.0000E + 00**
	ELAPO2	3.7860E − 45	1.1357E − 43	4.8673E − 43	1.4768E − 43
F11	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	5.5187E − 111	1.5603E − 107	7.9877E − 107	2.6781E − 107

Table 12: Statistical results for unimodal benchmark functions of different ELAPO (n = 100).

Function	Algorithm	Min.	Mean	Max.	Std
F1	ELAPO	**1.2060E − 201**	**5.2007E − 193**	**4.8628E − 192**	**0.0000E + 00**
	ELAPO1	9.1470E − 188	1.2551E − 185	8.4746E − 185	**0.0000E + 00**
	ELAPO2	1.3352E − 42	7.5917E − 41	5.2215E − 40	1.5892E − 40
F2	ELAPO	**6.6667E − 01**	**6.6667E − 01**	**6.6667E − 01**	**1.1703E − 16**
	ELAPO1	**6.6667E − 01**	**6.6667E − 01**	**6.6667E − 01**	4.6472E − 08
	ELAPO2	**6.6667E − 01**	**6.6667E − 01**	**6.6667E − 01**	**1.1703E − 16**
F3	ELAPO	**−1.0000E + 00**	**−1.0000E + 00**	**−1.0000E + 00**	**0.0000E + 00**
	ELAPO1	**−1.0000E + 00**	**−1.0000E + 00**	**−1.0000E + 00**	**0.0000E + 00**
	ELAPO2	**−1.0000E + 00**	**−1.0000E + 00**	**−1.0000E + 00**	**0.0000E + 00**
F4	ELAPO	**2.5939E − 179**	**3.5068E − 174**	**2.9799E − 173**	**0.0000E + 00**
	ELAPO1	2.2028E − 176	2.5228E − 171	2.4291E − 170	**0.0000E + 00**
	ELAPO2	2.3879E − 38	2.6197E − 25	2.6197E − 24	8.2841E − 25
F5	ELAPO	**1.6572E − 06**	**6.9030E − 05**	4.1043E − 04	1.2101E − 04
	ELAPO1	4.9616E − 05	1.3763E − 04	**2.8369E − 04**	**8.4291E − 05**
	ELAPO2	9.1224E − 05	2.5213E − 04	5.0008E − 04	1.3659E − 04
F6	ELAPO	7.5408E + 01	**7.8513E + 01**	**8.1196E + 01**	**1.6552E + 00**
	ELAPO1	9.2775E + 01	9.3408E + 01	9.4160E + 01	4.6870E − 01
	ELAPO2	**2.5737E − 03**	8.2837E + 01	1.8433E + 02	4.3749E + 01
F7	ELAPO	**5.8216E − 187**	**1.1968E − 182**	**8.1321E − 182**	**0.0000E + 00**
	ELAPO1	3.1789E − 181	2.5647E − 178	9.2747E − 178	**0.0000E + 00**
	ELAPO2	2.7609E − 39	2.9492E − 36	2.7378E − 35	8.5881E − 36
F8	ELAPO	1.2973E − 79	**4.0687E − 78**	**1.3134E − 77**	**3.8556E − 78**
	ELAPO1	**9.2722E − 80**	5.4812E − 78	2.3545E − 77	7.5349E − 78
	ELAPO2	1.1764E − 12	6.5701E − 12	2.2367E − 11	6.4475E − 12
F9	ELAPO	**5.4476E − 100**	**1.7153E − 97**	**8.1258E − 97**	**2.6804E − 97**
	ELAPO1	2.1502E − 97	1.1376E − 94	7.0599E − 94	2.1352E − 94
	ELAPO2	1.4540E − 22	1.1495E − 21	2.7186E − 21	9.4578E − 22
F10	ELAPO	**4.8377E − 192**	**2.1712E − 185**	**1.6637E − 184**	**0.0000E + 00**
	ELAPO1	1.7640E − 181	1.7228E − 178	7.7291E − 178	**0.0000E + 00**
	ELAPO2	7.6665E − 42	2.0437E − 40	1.2371E − 39	3.8039E − 40
F11	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	6.2316E − 111	2.4731E − 105	1.4424E − 104	4.8050E − 105

Table 13: Statistical results for multimodal benchmark functions of different ELAPO (n = 30).

Function	Algorithm	Min.	Mean	Max.	Std
F12	ELAPO	**8.8818E − 16**	**8.8818E − 16**	**8.8818E − 16**	**0.0000E + 00**
	ELAPO1	**8.8818E − 16**	**8.8818E − 16**	**8.8818E − 16**	**0.0000E + 00**
	ELAPO2	**8.8818E − 16**	**8.8818E − 16**	**8.8818E − 16**	**0.0000E + 00**
F13	ELAPO	**7.4429E − 107**	**6.1910E − 102**	**5.5536E − 101**	**1.7386E − 101**
	ELAPO1	1.4046E − 100	8.2536E − 100	1.6045E − 99	4.1699E − 100
	ELAPO2	3.4320E − 19	3.5435E − 04	1.9397E − 03	6.1675E − 04
F14	ELAPO	**7.0864E − 50**	**1.4029E − 48**	**5.2017E − 48**	**1.7605E − 48**
	ELAPO1	5.9353E − 47	1.4937E − 46	2.8483E − 46	6.9910E − 47
	ELAPO2	2.1593E − 10	4.4564E − 10	7.7516E − 10	1.7974E − 10
F15	ELAPO	9.0744E − 01	**1.4563E + 00**	**2.0765E + 00**	**4.3965E − 01**
	ELAPO1	8.3340E + 00	9.5357E + 00	1.0784E + 01	8.6499E − 01
	ELAPO2	**3.3712E − 01**	2.5937E + 00	1.0258E + 01	3.4500E + 00
F16	ELAPO	**1.5705E − 32**	**8.4188E − 25**	**4.2955E − 24**	**1.5385E − 24**
	ELAPO1	1.5184E − 17	7.2242E − 17	2.2432E − 16	6.2807E − 17
	ELAPO2	**1.5705E − 32**	7.6947E − 05	7.6947E − 04	2.4333E − 04
F17	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	3.2209E − 02	1.0586E − 01	4.3686E − 02

Table 13: Continued.

Function	Algorithm	Min.	Mean	Max.	Std
F18	ELAPO	**−2.9000E + 01**	**−2.9000E + 01**	**−2.9000E + 01**	**0.0000E + 00**
	ELAPO1	**−2.9000E + 01**	**−2.9000E + 01**	**−2.9000E + 01**	**0.0000E + 00**
	ELAPO2	−2.4010E + 01	−2.0005E + 01	−9.7242E + 00	5.3079E + 00
F19	ELAPO	−4.9123E + 03	**−4.8710E + 03**	**−4.8355E + 03**	**2.4102E + 01**
	ELAPO1	−4.9107E + 03	−4.7477E + 03	−4.3972E + 03	1.6214E + 02
	ELAPO2	**−4.9239E + 03**	−4.8603E + 03	−4.7473E + 03	5.2864E + 01
F20	ELAPO	**7.1841E − 05**	**2.0743E − 03**	**6.5185E − 03**	**2.6236E − 03**
	ELAPO1	1.6150E − 03	1.5982E − 02	4.3113E − 02	1.2385E − 02
	ELAPO2	1.1119E − 04	3.4940E − 03	1.2353E − 02	4.8505E − 03
F21	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
F22	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
F23	ELAPO	**5.8304E − 90**	**7.9899E − 02**	**9.9873E − 02**	4.2110E − 02
	ELAPO1	3.3769E − 85	8.9886E − 02	**9.9873E − 02**	3.1583E − 02
	ELAPO2	9.9873E − 02	9.9873E − 02	**9.9873E − 02**	**5.2771E − 13**
F24	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
F25	ELAPO	3.0957E + 02	3.4556E + 02	3.9799E + 02	3.3722E + 01
	ELAPO1	3.8740E + 02	4.0865E + 02	4.1395E + 02	**1.1183E + 01**
	ELAPO2	**4.0677E + 00**	**7.4112E + 01**	**2.4969E + 02**	7.3468E + 01

Table 14: Statistical results for multimodal benchmark functions of different ELAPO (n = 100).

Function	Algorithm	Min.	Mean	Max.	Std
F12	ELAPO	**8.8818E − 16**	**8.8818E − 16**	**8.8818E − 16**	**0.0000E + 00**
	ELAPO1	**8.8818E − 16**	**8.8818E − 16**	**8.8818E − 16**	**0.0000E + 00**
	ELAPO2	**8.8818E − 16**	**8.8818E − 16**	**8.8818E − 16**	**0.0000E + 00**
F13	ELAPO	**8.7095E − 103**	**3.6083E − 99**	**1.8257E − 98**	**5.4747E − 99**
	ELAPO1	2.5297E − 98	2.0018E − 96	1.4412E − 95	4.4005E − 96
	ELAPO2	4.9659E − 04	1.0832E − 02	4.9968E − 02	1.6131E − 02
F14	ELAPO	**2.0856E − 46**	**1.7929E − 44**	**9.4887E − 44**	**2.8722E − 44**
	ELAPO1	3.9777E − 45	4.6572E − 44	1.1290E − 43	3.4345E − 44
	ELAPO2	1.5153E − 08	7.9214E − 08	3.0473E − 07	8.5298E − 08
F15	ELAPO	**3.1824E + 00**	**7.2183E + 00**	**3.2412E + 01**	8.8821E + 00
	ELAPO1	3.9427E + 01	4.0794E + 01	4.1923E + 01	**8.9874E − 01**
	ELAPO2	3.3002E + 00	1.7828E + 01	4.0400E + 01	1.7857E + 01
F16	ELAPO	**4.7116E − 33**	**4.7116E − 33**	**4.7116E − 33**	**1.9082E − 48**
	ELAPO1	7.3562E − 05	1.4325E − 04	2.6188E − 04	5.4182E − 05
	ELAPO2	**4.7116E − 33**	3.1101E − 03	3.1101E − 02	9.8349E − 03
F17	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	1.0827E − 02	5.1706E − 02	1.7751E − 02
F18	ELAPO	**−9.9000E + 01**	**−7.4063E + 01**	−2.6004E + 01	3.2165E + 01
	ELAPO1	**−9.9000E + 01**	−2.3413E + 01	−1.2547E + 01	2.6599E + 01
	ELAPO2	−8.0137E + 01	−6.3474E + 01	**−2.8524E + 01**	**2.0801E + 01**
F19	ELAPO	−1.3042E + 05	−6.5750E + 04	−9.5402E + 02	4.1853E + 04
	ELAPO1	−3.0396E + 03	−2.5921E + 03	−2.1467E + 03	**2.2698E + 02**
	ELAPO2	**−1.3365E + 05**	**−7.7752E + 04**	**−3.4716E + 04**	3.3300E + 04
F20	ELAPO	**3.4881E − 04**	**7.3630E − 02**	**4.5528E − 01**	**1.5569E − 01**
	ELAPO1	8.5545E − 01	1.6562E + 00	2.0958E + 00	3.8738E − 01
	ELAPO2	9.3205E − 04	1.2621E − 01	5.5200E − 01	1.7669E − 01

Table 14: Continued.

Function	Algorithm	Min.	Mean	Max.	Std
F21	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
F22	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
F23	ELAPO	**9.9873E − 02**	**9.9873E − 02**	**9.9873E − 02**	**7.7820E − 17**
	ELAPO1	**9.9873E − 02**	**9.9873E − 02**	**9.9873E − 02**	9.4568E − 11
	ELAPO2	**9.9873E − 02**	**9.9873E − 02**	**9.9873E − 02**	9.6023E − 16
F24	ELAPO	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO1	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
	ELAPO2	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**	**0.0000E + 00**
F25	ELAPO	1.5069E + 03	3.6192E + 03	4.2156E + 03	8.1059E + 02
	ELAPO1	4.5995E + 03	4.5995E + 03	4.5995E + 03	**9.5869E − 13**
	ELAPO2	**2.3949E + 01**	**1.0427E + 03**	**3.1627E + 03**	1.1139E + 03

Table 15: Statistical results for CEC 2014 benchmark functions of different ELAPO (n = 30).

Function	Algorithm	Min.	Mean	Max.	Std
F26	ELAPO	1.7031E + 05	**4.4635E + 05**	**1.4637E + 06**	**4.1890E + 05**
	ELAPO1	1.3255E + 05	7.8737E + 05	1.6349E + 06	5.4983E + 05
	ELAPO2	**8.6131E + 04**	6.9261E + 05	2.3688E + 06	6.8875E + 05
F27	ELAPO	2.0002E + 02	**2.0030E + 02**	**2.0173E + 02**	**5.2289E − 01**
	ELAPO1	2.1032E + 02	4.5935E + 02	9.2097E + 02	2.0827E + 02
	ELAPO2	**2.0000E + 02**	2.0045E + 02	2.0248E + 02	7.6958E − 01
F28	ELAPO	**4.0001E + 02**	**4.0704E + 02**	**4.6723E + 02**	2.1148E + 01
	ELAPO1	4.0408E + 02	4.6828E + 02	5.6357E + 02	4.7798E + 01
	ELAPO2	**4.0001E + 02**	4.0766E + 02	4.6765E + 02	**2.1129E + 01**
F29	ELAPO	1.3442E + 04	**1.6246E + 05**	**3.3345E + 05**	**1.0464E + 05**
	ELAPO1	1.0292E + 05	2.6555E + 05	6.1334E + 05	2.0477E + 05
	ELAPO2	**6.9968E + 03**	4.2214E + 05	1.5059E + 06	5.2014E + 05
F30	ELAPO	**2.6152E + 03**	**2.6152E + 03**	**2.6152E + 03**	**4.7935E − 13**
	ELAPO1	**2.6152E + 03**	**2.6152E + 03**	**2.6152E + 03**	1.2234E − 11
	ELAPO2	**2.6152E + 03**	**2.6152E + 03**	**2.6152E + 03**	3.6190E − 12
F31	ELAPO	**2.6000E + 03**	**2.6000E + 03**	**2.6000E + 03**	3.5666E − 04
	ELAPO1	**2.6000E + 03**	**2.6000E + 03**	**2.6000E + 03**	**1.7811E − 04**
	ELAPO2	**2.6000E + 03**	**2.6000E + 03**	**2.6000E + 03**	1.5511E − 03
F32	ELAPO	**2.7000E + 03**	**2.7000E + 03**	**2.7000E + 03**	**0.0000E + 00**
	ELAPO1	**2.7000E + 03**	**2.7000E + 03**	**2.7000E + 03**	**0.0000E + 00**
	ELAPO2	**2.7000E + 03**	**2.7000E + 03**	**2.7000E + 03**	**0.0000E + 00**

ELAPO1 and ELAPO2. This phenomenon is expected and reasonable according to the "no free lunch" (NFL) theorem [59] that stated that no any single algorithm is able to efficiently solve all optimization problems. For some other multimodal functions, the two strategies are sensitive to the number of dimensions. For instance, the contribution of F18 mainly comes from the quasi-opposition-based learning strategy under the condition at n = 30, while at n = 100 the main contribution is from the dimensional search strategy. For F19, ELAPO1 seems to have equal contribution as ELAPO2 at n = 30, while at n = 100, the former tends to have negative effect on the global search performance.

The statistical results for CEC 2014 benchmark functions are presented in Tables 15 and 16. It can be seen from the tables that both strategies have positive influences on F26, F27, F28, and F29, and the dimensional search strategy tends to take greater effects especially when the number of dimensions is higher. For composite functions, as the complexity of the function increases, the two strategies are still able to make slight contributions, and thus, as per Tables 9 and 10, ELAPO can get competitive results over all other algorithms.

In summary, equipping with any individual strategy only is insufficient to achieve the desired results, but integrating the two strategies results in excellent performance for most of benchmark functions. This superior performance of ELAPO verifies its appropriate taking care of the exploration-exploitation trade-off problem with the introduction of the proposed two strategies.

Table16: Statistical results for CEC 2014 benchmark functions of different ELAPO ($n = 100$).

Function	Algorithm	Min.	Mean	Max.	Std
F26	ELAPO	2.4634$E+06$	**5.2161E+06**	**1.3141E+07**	**3.1061E+06**
	ELAPO1	2.2387$E+07$	4.5695$E+07$	6.2258$E+07$	1.4119$E+07$
	ELAPO2	**2.4614E+06**	7.0362$E+06$	1.5330$E+07$	4.6938$E+06$
F27	ELAPO	**6.2568E+02**	**7.8528E+03**	**1.6154E+04**	**4.6668E+03**
	ELAPO1	9.6085$E+07$	2.1771$E+08$	3.3325$E+08$	6.8438$E+07$
	ELAPO2	1.4145$E+03$	1.0199$E+04$	2.4210$E+04$	7.4670$E+03$
F28	ELAPO	4.8012$E+02$	**5.4633E+02**	**6.3713E+02**	**4.9530E+01**
	ELAPO1	7.2188$E+02$	8.8310$E+02$	9.4605$E+02$	7.5816$E+01$
	ELAPO2	**4.0910E+02**	5.6801$E+02$	6.3932$E+02$	6.6571$E+01$
F29	ELAPO	8.6796$E+05$	**2.0363E+06**	**5.4231E+06**	**1.2767E+06**
	ELAPO1	1.2845$E+06$	3.6401$E+06$	7.6576$E+06$	2.5212$E+06$
	ELAPO2	**7.0505E+05**	2.1188$E+06$	6.4199$E+06$	1.7726$E+06$
F30	ELAPO	**2.5000E+03**	2.6451$E+03$	2.7624$E+03$	**6.2281E+01**
	ELAPO1	**2.5000E+03**	2.7849$E+03$	3.4244$E+03$	3.5275$E+02$
	ELAPO2	**2.5000E+03**	**2.6186E+03**	**2.6483E+03**	6.2509$E+01$
F31	ELAPO	**2.6000E+03**	**2.6000E+03**	**2.6000E+03**	1.0968$E-03$
	ELAPO1	**2.6000E+03**	**2.6000E+03**	**2.6000E+03**	**6.0886E-04**
	ELAPO2	**2.6000E+03**	**2.6000E+03**	**2.6000E+03**	4.5087$E-03$
F32	ELAPO	**2.7000E+03**	**2.7000E+03**	**2.7000E+03**	**0.0000E+00**
	ELAPO1	**2.7000E+03**	**2.7000E+03**	**2.7000E+03**	**0.0000E+00**
	ELAPO2	**2.7000E+03**	**2.7000E+03**	**2.7000E+03**	**0.0000E+00**

Table 17: Results of Wilcoxon's test for ELAPO against other six algorithms for each benchmark function with 10 independent runs at $n = 30$ ($\alpha = 0.05$).

Function	LAPO vs. ELAPO		SSA vs. ELAPO		Jaya vs. ELAPO		IBB-BC vs. ELAPO		ODE1 vs. ELAPO		ALO vs. ELAPO	
	p value	Win	p value	Win	p value	Win	p value	Win	p value	Win	p value	Win
F1	6.5157$E-02$	−	3.8030$E-05$	+	8.8057$E-06$	+	1.8054$E-02$	+	3.3181$E-01$	−	1.0788$E-01$	−
F2	1.1430$E-03$	+	3.5370$E-01$	−	8.2032$E-02$	−	2.0905$E-01$	−	1.6587$E-01$	−	4.1164$E-02$	+
F3	—	=	1.4851$E-04$	+	4.8142$E-06$	+	2.1981$E-17$	+	1.4807$E-02$	+	8.7627$E-05$	+
F4	2.3713$E-01$	−	6.8108$E-03$	+	1.2111$E-05$	+	1.1967$E-04$	+	1.3930$E-01$	−	1.6670$E-04$	+
F5	3.2636$E-03$	+	1.1738$E-06$	+	2.1936$E-09$	+	6.6659$E-09$	+	6.8124$E-07$	+	1.3194$E-05$	+
F6	3.9780$E-11$	+	5.6264$E-04$	+	1.4458$E-04$	+	4.7296$E-02$	+	3.8957$E-04$	+	1.3327$E-02$	+
F7	1.3127$E-02$	+	4.2515$E-04$	+	2.6918$E-05$	+	4.0856$E-02$	+	2.0166$E-01$	−	2.4269$E-02$	+
F8	1.0554$E-03$	+	2.6806$E-07$	+	1.3055$E-04$	+	3.2266$E-03$	+	6.0020$E-05$	+	1.4067$E-06$	+
F9	1.7774$E-01$	−	1.7559$E-06$	+	3.1821$E-06$	+	4.0316$E-04$	+	1.8318$E-05$	+	4.8798$E-02$	+
F10	6.0410$E-02$	−	4.0552$E-03$	+	3.8508$E-06$	+	3.2237$E-05$	+	1.1459$E-02$	+	4.8526$E-03$	+
F11	2.5412$E-01$	−	2.3083$E-01$	−	1.8567$E-01$	−	1.6998$E-02$	+	2.2426$E-01$	−	9.9079$E-04$	+
F12	5.7296$E-03$	+	1.1192$E-02$	+	4.6725$E-03$	+	3.7373$E-14$	+	3.2594$E-01$	−	8.0055$E-05$	+
F13	1.8191$E-03$	+	1.8814$E-04$	+	8.9399$E-05$	+	2.5553$E-03$	+	1.8822$E-01$	−	5.4968$E-04$	+
F14	9.5904$E-04$	+	8.1454$E-07$	+	4.6052$E-08$	+	6.4579$E-09$	+	6.2990$E-07$	+	6.7484$E-09$	+
F15	1.3848$E-10$	+	1.2732$E-03$	+	3.4198$E-13$	+	3.2496$E-12$	+	1.9113$E-13$	+	7.5973$E-11$	+
F16	3.4344$E-01$	−	1.3527$E-03$	+	1.9997$E-05$	+	3.6772$E-02$	+	2.9349$E-02$	+	1.0442$E-05$	+
F17	3.4344$E-01$	−	6.7604$E-04$	+	2.1918$E-03$	+	4.1350$E-02$	+	2.0726$E-02$	+	1.0132$E-02$	+
F18	1.3100$E-05$	+	1.2328$E-06$	+	3.2377$E-14$	+	2.2121$E-17$	+	7.6070$E-14$	+	3.5169$E-09$	+
F19	2.9586$E-04$	+	3.4911$E-03$	+	7.4745$E-02$	−	9.9030$E-03$	+	1.7725$E-05$	+	7.2303$E-06$	+
F20	2.8427$E-02$	+	6.2178$E-02$	+	4.0529$E-05$	+	1.7734$E-01$	−	1.1848$E-01$	−	8.2189$E-04$	+
F21	—	=	7.0957$E-02$	+	6.6615$E-11$	+	4.8172$E-06$	+	1.5624$E-08$	+	4.3379$E-06$	+
F22	1.6787$E-01$	−	8.1058$E-02$	+	6.4191$E-12$	+	1.2747$E-06$	+	7.1310$E-06$	+	4.4267$E-04$	+
F23	1.6785$E-01$	−	3.4082$E-10$	+	5.7993$E-11$	+	3.9727$E-07$	+	1.0620$E-05$	+	2.6960$E-08$	+
F24	—	=	2.9637$E-05$	+	5.0207$E-05$	+	1.7982$E-06$	+	1.6055$E-01$	−	4.2089$E-08$	+
F25	2.7440$E-04$	+	6.8708$E-01$	−	9.8070$E-12$	+	9.7584$E-12$	+	2.6606$E-08$	+	1.6495$E-09$	+
F26	6.1439$E-02$	−	7.0291$E-04$	+	1.3762$E-05$	+	3.8405$E-03$	+	6.1551$E-03$	+	5.0196$E-05$	+
F27	6.8071$E-03$	+	7.0180$E-03$	+	1.5724$E-09$	+	1.7808$E-04$	+	5.2722$E-02$	−	5.0467$E-04$	+
F28	3.6344$E-03$	+	4.8441$E-05$	+	3.0928$E-08$	+	3.2500$E-09$	+	1.7440$E-07$	+	1.9483$E-05$	+
F29	4.6908$E-02$	+	4.9702$E-04$	+	1.1193$E-04$	+	1.0340$E-02$	+	3.2293$E-03$	+	4.3079$E-03$	+
F30	2.5163$E-02$	+	2.6125$E-02$	+	2.8404$E-07$	+	3.5374$E-05$	+	3.2647$E-01$	−	1.3533$E-04$	+
F31	2.6889$E-07$	+	4.1868$E-07$	+	3.8867$E-09$	+	2.0918$E-10$	+	4.8250$E-08$	+	2.7003$E-10$	+
F32	1.6785$E-01$	−	6.3084$E-04$	+	1.5043$E-09$	+	4.1352$E-07$	+	2.5620$E-10$	+	8.4991$E-08$	+
+/−	—	18/11	—	26/6	—	29/3		30/2		21/11		31/1

Table 18: Results of Wilcoxon's test for SSA against other six algorithms for each benchmark function with 10 independent runs at $n = 100$ ($\alpha = 0.05$).

Function	LAPO vs. ELAPO		SSA vs. ELAPO		Jaya vs. ELAPO		IBB-BC vs. ELAPO		ODE1 vs. ELAPO		ALO vs. ELAPO	
	p value	Win	p value	Win	p value	Win	p value	Win	p value	Win	p value	Win
F1	$1.5770E-01$	−	$4.2350E-08$	+	$1.5231E-07$	+	$6.3386E-04$	+	$1.3419E-03$	+	$3.5682E-06$	+
F2	$7.6218E-03$	+	$6.0367E-08$	+	$1.9065E-04$	+	$9.3693E-03$	+	$3.2161E-05$	+	$5.1856E-07$	+
F3	—	=	$3.5796E-08$	+	$2.2802E-07$	+	$4.4140E-54$	+	$2.3346E-02$	+	$3.0033E-05$	+
F4	$1.3041E-01$	−	$3.5293E-08$	+	$2.8240E-04$	+	$1.4775E-06$	+	$2.7093E-02$	+	$9.4948E-07$	+
F5	$5.6333E-03$	+	$1.0084E-08$	+	$4.8518E-06$	+	$1.3531E-06$	+	$9.5032E-05$	+	$6.7646E-10$	+
F6	$9.5047E-12$	+	$3.3028E-06$	+	$1.8052E-05$	+	$2.2892E-03$	+	$2.4625E-02$	+	$8.5749E-03$	+
F7	$1.4156E-01$	−	$3.0753E-08$	+	$6.9812E-08$	+	$3.8161E-04$	+	$2.0774E-03$	+	$1.8206E-06$	+
F8	$1.9728E-04$	+	$1.3132E-12$	+	$5.7059E-11$	+	$5.6270E-11$	+	$2.1460E-08$	+	$2.7677E-09$	+
F9	$2.7143E-02$	+	$4.6832E-11$	+	$3.2634E-07$	+	$2.7294E-10$	+	$5.0914E-05$	+	$2.2988E-03$	+
F10	$4.3062E-02$	+	$4.2546E-08$	+	$1.6382E-06$	+	$8.1021E-03$	+	$9.3687E-02$	−	$1.0917E-04$	+
F11	$2.6996E-01$	−	$2.2696E-01$	−	$5.0667E-02$	−	$1.1055E-03$	+	$3.3891E-01$	−	$1.2744E-03$	+
F12	$7.6477E-02$	−	$5.8330E-05$	+	$3.0288E-11$	+	$4.6170E-19$	+	$8.3106E-07$	+	$6.0946E-08$	+
F13	$1.0360E-03$	+	$5.5028E-07$	+	$9.0010E-09$	+	$5.9879E-09$	+	$1.9392E-03$	+	$2.4309E-06$	+
F14	$3.8838E-04$	+	$9.7111E-10$	+	$1.2974E-11$	+	$9.7901E-14$	+	$4.5811E-09$	+	$3.3538E-11$	+
F15	$8.3808E-07$	+	$9.8135E-05$	+	$2.3964E-07$	+	$3.5313E-07$	+	$1.6716E-07$	+	$1.0160E-06$	+
F16	$3.1494E-01$	−	$1.4591E-05$	+	$9.7572E-02$	−	$1.4803E-03$	+	$8.8968E-02$	−	$5.9325E-06$	+
F17	—	=	$1.1073E-09$	+	$4.1210E-12$	+	$1.6710E-08$	+	$2.9375E-05$	+	$1.7779E-07$	+
F18	$3.9961E-04$	+	$6.4253E-01$	−	$1.7095E-04$	+	$8.8954E-05$	+	$2.0868E-04$	+	$1.9220E-03$	+
F19	$9.8149E-04$	+	$1.1788E-06$	+	$4.8892E-07$	+	$2.1151E-05$	+	$2.0236E-06$	+	$1.0299E-02$	+
F20	$4.3629E-07$	+	$5.7602E-01$	−	$6.2070E-09$	+	$1.1723E-02$	+	$6.6614E-02$	−	$2.9847E-04$	+
F21	—	=	$3.5828E-11$	+	$2.7658E-12$	+	$7.1919E-11$	+	$7.3339E-10$	+	$1.1238E-07$	+
F22	—	=	$7.0366E-09$	+	$3.4023E-12$	+	$3.3868E-10$	+	$2.6364E-12$	+	$7.8276E-07$	+
F23	$5.2427E-02$	−	$2.3972E-10$	+	$1.0601E-11$	+	$2.3959E-12$	+	$4.1327E-09$	+	$4.4496E-09$	+
F24	—	=	$1.4813E-09$	+	$1.4883E-11$	+	$2.4103E-13$	+	$1.3202E-08$	+	$1.0014E-11$	+
F25	$4.0643E-03$	+	$1.3414E-09$	+	$1.9271E-04$	+	$3.5879E-09$	+	$1.1709E-02$	+	$6.5374E-10$	+
F26	$9.1228E-07$	+	$2.4299E-06$	+	$4.8229E-08$	+	$2.4166E-07$	+	$2.9383E-05$	+	$9.1708E-07$	+
F27	$1.9144E-02$	+	$8.2124E-08$	+	$3.2662E-10$	+	$2.7758E-04$	+	$6.3573E-03$	+	$9.5400E-04$	+
F28	$2.7429E-07$	+	$2.3276E-07$	+	$1.2105E-07$	+	$1.2547E-07$	+	$1.7003E-06$	+	$9.3832E-09$	+
F29	$5.8224E-02$	−	$1.1899E-03$	+	$1.0274E-09$	+	$2.5110E-03$	+	$2.2029E-05$	+	$4.0114E-04$	+
F30	$7.7749E-01$	−	$7.1957E-01$	−	$2.2963E-07$	+	$6.5996E-07$	+	$7.2799E-01$	−	$1.1532E-04$	+
F31	$2.3273E-05$	+	$3.4931E-17$	+	$4.5281E-12$	+	$2.0453E-13$	+	$7.3367E-16$	+	$1.4373E-09$	+
F32	$1.0442E-02$	+	$8.3154E-07$	+	$4.3080E-10$	+	$1.0525E-08$	+	$1.7940E-09$	+	$4.9109E-10$	+
+/−	—	18/9	—	28/4	—	30/2	—	32/0	—	27/5	—	32/0

Table 19: Friedman ranks for each benchmark function of all algorithms at $n = 30$.

Fun	ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F1	1	2	4	5	6	3	7
F2	1.5	1.5	4	5	7	3	6
F3	2	2	5	6	7	2	4
F4	1	2	4	5	7	3	6
F5	1	2	7	6	3	4	5
F6	1	2	4	5	7	3	6
F7	1	2	4	5	6	3	7
F8	1	2	4	5	3	6	7
F9	1	2	4	6	5	3	7
F10	1	2	5	7	6	3	4
F11	1	2	3	4	6	5	7
F12	1	2	4	5	7	3	6
F13	1	2	3	7	5	4	6
F14	1	2	4	5	6	3	7
F15	2	3	1	6	7	5	4
F16	1	3	2	6	4	5	7
F17	1	2	5	7	6	3	4
F18	1	3	2	6	7	5	4

Table 19: Continued.

Fun	ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F19	1	2	6	4	7	5	3
F20	2	6	1	7	3	4	5
F21	1.5	1.5	3	7	4	6	5
F22	1	3	2	7	4	6	5
F23	1	2	5	6	7	3	4
F24	1.5	1.5	4	5	6	3	7
F25	1	3	2	6	7	5	4
F26	1	2	4	7	5	3	6
F27	1	2	6	7	4	3	5
F28	1	2	4	7	6	3	5
F29	2	3	6	7	4	1	5
F30	2	2	2	6	7	3	5
F31	1.5	1.5	4	6	7	3	5
F32	1.5	1.5	4	6	7	3	5
Average	1.234375	2.234375	3.8125	5.90625	5.71875	3.65625	5.40625
Rank	1	2	4	7	6	3	5

Table 20: Friedman ranks for each benchmark function of all algorithms at $n = 100$.

Fun	ELAPO	LAPO	SSA	Jaya	IBB-BC	ODE1	ALO
F1	1	2	3	6	7	4	5
F2	1.5	1.5	5	7	3	6	4
F3	2	2	4	6	7	5	2
F4	1	2	3	5	7	4	6
F5	1	2	6	7	3	4	5
F6	1	2	5	7	3	6	4
F7	1	2	3	6	7	4	5
F8	1	2	7	5	6	3	4
F9	1	2	4	5	6	3	7
F10	1	2	4	7	6	5	3
F11	1	2	3	7	5	4	6
F12	1	2	6	5	7	3	4
F13	1	2	4	7	6	3	5
F14	1	2	4	6	7	3	5
F15	1	3	2	7	5	6	4
F16	1	2	4	7	3	6	5
F17	1.5	1.5	5	6	7	4	3
F18	1	4	2	6	7	5	3
F19	1	3	6	5	7	4	2
F20	2	6	1	7	4	3	5
F21	1.5	1.5	3	7	5	6	4
F22	1.5	1.5	3	7	5	6	4
F23	1.5	1.5	3	6	7	4	5
F24	1.5	1.5	4	5	7	3	6
F25	1	2	4	7	3	6	5
F26	1	2	5	7	6	3	4
F27	1	5	2	7	6	4	3
F28	1	4	2	7	6	3	5
F29	1	2	6	7	4	5	3
F30	1	2	4	7	6	3	5
F31	1.5	1.5	3	6	7	4	5
F32	1.5	1.5	4	7	6	3	5
Average	1.1875	2.28125	3.875	6.375	5.65625	4.21875	4.40625
Rank	1	2	3	7	6	4	5

Table 21: Ranking of algorithms using MAE.

Algorithm	MAE ($n = 30$)	Rank	MAE ($n = 100$)	Rank
ELAPO	$1.9004E+04$	1	$2.3027E+05$	1
LAPO	$3.6355E+04$	2	$2.8935E+05$	2
ODE1	$7.8336E+04$	3	$1.3912E+07$	5
SSA	$2.7223E+05$	4	$1.0381E+07$	3
IBB-BC	$4.4013E+05$	5	$9.3375E+08$	6
ALO	$4.7895E+05$	6	$1.3480E+07$	4
Jaya	$2.1949E+08$	7	$4.1882E+09$	7

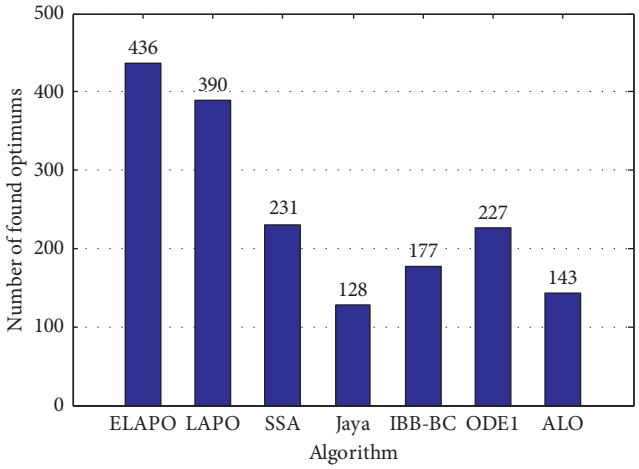

Figure 7: Comparison of algorithms in finding the global optimal solution out of 640 runs.

4.5. Statistical Analysis. In order to analyze the performance of any two algorithms, the Wilcoxon signed-rank test and Friedman test [63] are considered for the present work. The results of Wilcoxon's test for ELAPO against other six algorithms are summarized in Tables 17 and 18 for $n = 30$ and 100, respectively. The test is carried out by considering the best solution of each algorithm on each benchmark function with 10 independent runs and a significance level of $\alpha = 0.05$. In Tables 17 and 18, "+" sign indicates that the reference algorithm outperforms the compared one, "−" sign indicates that the reference algorithm is inferior to the compared one, and "=" sign indicates that both algorithms have comparable performances. The results of last row of the tables show that the proposed ELAPO has a larger number of "+" counts in comparison to other algorithms, confirming that ELAPO is better than the other six compared algorithms under 95% level of significance. The results of the Friedman test are presented in Tables 19 and 20 for $n = 30$ and 100, respectively. The last row of the tables depicts the ranks computed through the Friedman test. As can be seen in the table, ELAPO is the best performing algorithm of seven optimization algorithms.

The quantitative analysis is also carried out for seven algorithms with an index of mean absolute error (MAE) which is an effective performance index for ranking the optimization algorithms and is defined by [64]

$$\text{MAE} = \frac{\sum_{j=1}^{N} \left| m_j - k_j \right|}{N}, \tag{15}$$

where m_j is the mean of optimal values, k_j is the actual global optimal value, and N is the number of samples. In the present work, N is the number of benchmark functions. The MAE of all algorithms and their ranking for all functions are given in Table 19. It is clear to find that ELAPO ranks No. 1 and provides the minimum MAE in all cases. ELAPO reaches the optimum solution 436 times out of 640 runs (10 runs for each test function for $n = 30$ and 100, respectively) and comes in the first rank as shown in Figure 7. It is concluded that ELAPO provides the best performance in comparison to other six optimization algorithms.

5. Conclusions

In this paper, an enhanced lightning attachment procedure optimization called ELAPO is proposed for global optimization problems. The exploration and exploitation abilities of the basic LAPO are appropriately balanced in the search process. The quasi-opposition-based learning strategy is applied to control the convergence speed and to improve both exploration and exploitation abilities of the algorithm. To further enhance the exploitation capability, the dimensional search strategy is employed, which inherits the good information from the best solution in each iteration and thus increases the convergence precision of the proposed algorithm. The efficiency of ELAPO is examined on unimodal, multimodal, and CEC 2014 benchmark functions. The statistic results show that the proposed algorithm has superior performance in terms of accuracy and convergence rates when compared with other six state-of-the-art algorithms including LAPO, SSA, Jaya, IBB-BC, ODE1, and ALO.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by a research grant from the National Key Research and Development Program of China (project no. 2017YFC1500400) and the National Natural Science Foundation of China (51808147).

References

[1] D. Gong, J. Sun, and Z. Miao, "A set-based genetic algorithm for interval many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 47–60, 2018.

[2] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, vol. 5, Kluwer Academic Publishers, Hingham, MA, USA, 2006.

[3] P. P. Repoussis, C. D. Tarantilis, O. Bräysy, and G. Ioannou, "A hybrid evolution strategy for the open vehicle routing

problem," *Computers & Operations Research*, vol. 37, no. 3, pp. 443–455, 2010.

[4] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[5] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[6] C. F. Wang and K. Liu, "A novel particle swarm optimization algorithm for global optimization," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 9482073, 9 pages, 2016.

[7] D. Marco, D. O. M. A. Montes, O. Sabrina, and S. Thomas, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2007.

[8] D. Karaboga and B. Basturk, "An artificial bee colony (ABC) algorithm for numeric function optimization," in *IEEE Swarm Intelligence Symposium*, pp. 181–184, Indianapolis, IN, USA, May 2006.

[9] A. Ritthipakdee, A. Thammano, N. Premasathian, and D. Jitkongchuen, "Firefly mating algorithm for continuous optimization problems," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 8034573, 10 pages, 2017.

[10] H. Liu, F. Yi, and H. Yang, "Adaptive grouping cloud model shuffled frog leaping algorithm for solving continuous optimization problems," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 5675349, 8 pages, 2016.

[11] X. Lu and Y. Zhou, "A novel global convergence algorithm: bee collecting pollen algorithm," in *Proceedings of the International Conference on Intelligent Computing*, pp. 518–525, Springer, Shanghai, China, September 2008.

[12] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the 2009 World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 210–214, IEEE, Coimbatore, India, December 2009.

[13] Y. Shiqin, J. Jianjun, and Y. Guangxing, "A dolphin partner optimization," in *Proceedings of the 2009 WRI Global Congress on Intelligent Systems*, vol. 1, pp. 124–128, IEEE, Xiamen, China, May 2009.

[14] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., pp. 65–74, Springer, Berlin, Germany, 2010.

[15] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization," 2010, https://arxiv.org/abs/1003.1409.

[16] R. Oftadeh, M. J. Mahjoob, and M. Shariatpanahi, "A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search," *Computers & Mathematics with Applications*, vol. 60, no. 7, pp. 2087–2098, 2010.

[17] T. Zheng, J. Liu, W. Luo, and Z. Lu, "Structural damage identification using cloud model based fruit fly optimization algorithm," *Structural Engineering and Mechanics*, vol. 67, no. 3, pp. 245–254, 2018.

[18] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.

[19] S. A. Uymaz, G. Tezel, and E. Yel, "Artificial algae algorithm (AAA) for nonlinear global optimization," *Applied Soft Computing*, vol. 31, pp. 153–171, 2015.

[20] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[21] W. Yong, W. Tao, Z. Cheng-Zhi, and H. Hua-Juan, "A new stochastic optimization approach—dolphin swarm optimization algorithm," *International Journal of Computational Intelligence and Applications*, vol. 15, no. 2, article 1650011, 2016.

[22] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[23] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm," *Computers & Structures*, vol. 169, pp. 1–12, 2016.

[24] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: theory and application," *Advances in Engineering Software*, vol. 105, pp. 30–47, 2017.

[25] E. Jahani and M. Chizari, "Tackling global optimization problems with a novel algorithm—Mouth Brooding Fish algorithm," *Applied Soft Computing*, vol. 62, pp. 987–1002, 2018.

[26] G. Dhiman and V. Kumar, "Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications," *Advances in Engineering Software*, vol. 114, pp. 48–70, 2017.

[27] X. Qi, Y. Zhu, and H. Zhang, "A new meta-heuristic butterfly-inspired algorithm," *Journal of Computational Science*, vol. 23, pp. 226–239, 2017.

[28] T. Zheng and W. Luo, "An improved squirrel search algorithm for optimization," *Complexity*, vol. 2019, Article ID 6291968, 31 pages, 2019.

[29] B. Almonacid and R. Soto, "Andean Condor Algorithm for cell formation problems," *Natural Computing*, vol. 18, no. 2, pp. 351–381, 2019.

[30] N. A. Kallioras, N. D. Lagaros, and D. N. Avtzis, "Pity beetle algorithm—a new metaheuristic inspired by the behavior of bark beetles," *Advances in Engineering Software*, vol. 121, pp. 147–166, 2018.

[31] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269–283, 2008.

[32] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[33] O. K. Erol and I. Eksin, "A new optimization method: big Bang–Big Crunch," *Advances in Engineering Software*, vol. 37, no. 2, pp. 106–111, 2006.

[34] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, no. 3-4, pp. 267–289, 2010.

[35] A. Hatamlou, "Black hole: a new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175–184, 2013.

[36] R. A. Formato, "Central force optimization: a new deterministic gradient-like optimization metaheuristic," *Opsearch*, vol. 46, no. 1, pp. 25–51, 2009.

[37] H. Du, X. Wu, and J. Zhuang, "Small-world optimization algorithm for function optimization," in *Proceedings of the International Conference on Natural Computation*, pp. 264–273, Springer, Xi'an, China, September 2006.

[38] B. Alatas, "ACROA: artificial chemical reaction optimization algorithm for global optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13170–13180, 2011.

[39] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: ray optimization," *Computers & Structures*, vol. 112-113, pp. 283–294, 2012.

[40] H. Shah-Hosseini, "Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation," *International Journal of*

Computational Science and Engineering, vol. 6, no. 1-2, pp. 132–140, 2011.

[41] F. F. Moghaddam, R. F. Moghaddam, and M. Cheriet, "Curved space optimization: a random search based on general relativity theory," 2012, http://arxiv.org/abs/1208.2214.

[42] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: a nature-inspired algorithm for global optimization," *Neural Computing and Applications*, vol. 27, no. 2, pp. 495–513, 2016.

[43] Z. Zhang, J. Pan, W. Luo, K. R. Ramakrishnan, and H. K. Singh, "Vibration-based delamination detection in curved composite plates," *Composites Part A: Applied Science and Manufacturing*, vol. 119, pp. 261–274, 2019.

[44] A. F. Nematollahi, A. Rahiminejad, and B. Vahidi, "A novel physical based meta-heuristic optimization method known as Lightning Attachment Procedure Optimization," *Applied Soft Computing*, vol. 59, pp. 596–621, 2017.

[45] Y. Li, X. Zhao, Y. Wang, and M. Ren, "Multi-objective optimization of rolling schedules for tandem hot rolling based on opposition learning multi-objective genetic algorithm," in *Proceedings of the 2013 25th Chinese Control and Decision Conference (CCDC)*, pp. 846–849, IEEE, Guiyang, China, May 2013.

[46] K. Karthikeyan, S. Kannan, S. Baskar, and C. Thangaraj, "Application of opposition-based differential evolution algorithm to generation expansion planning problem," *Journal of Electrical Engineering and Technology*, vol. 8, no. 4, pp. 686–693, 2013.

[47] M. Y. Cheng and D. H. Tran, "Integrating chaotic initialized opposition multiple-objective differential evolution and stochastic simulation to optimize ready-mixed concrete truck dispatch schedule," *Journal of Management in Engineering*, vol. 32, no. 1, article 04015034, 2016.

[48] L. Yang, S. Xijia, and C. Deng, "Opposition-based learning particle swarm optimization of running gait for humanoid robot," *International Journal on Smart Sensing and Intelligent Systems*, vol. 8, no. 2, 2015.

[49] Z. Wu, Z. Ni, C. Zhang, and L. Gu, "A novel PSO for multi-stage portfolio planning," in *Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 4, pp. 71–77, IEEE, Shanghai, China, November 2009.

[50] Q. Xu, L. Guo, N. Wang, and Y. He, "COOBBO: a novel opposition-based soft computing algorithm for TSP problems," *Algorithms*, vol. 7, no. 4, pp. 663–684, 2014.

[51] S. K. Goudos, M. Deruyck, D. Plets, L. Martens, and W. Joseph, "Application of opposition-based learning concepts in reducing the power consumption in wireless access networks," in *Proceedings of the 2016 23rd International Conference on Telecommunications (ICT)*, pp. 1–5, IEEE, Thessaloniki, Greece, May 2016.

[52] A. Banerjee, V. Mukherjee, and S. P. Ghoshal, "An opposition-based harmony search algorithm for engineering optimization problems," *Ain Shams Engineering Journal*, vol. 5, no. 1, pp. 85–101, 2014.

[53] X. Z. Gao, J. Wang, J. M. Tanskanen, R. Bie, and P. Guo, "BP neural networks with harmony search method-based training for epileptic EEG signal classification," in *Proceedings of the 2012 Eighth International Conference on Computational Intelligence and Security*, pp. 252–257, IEEE, Guangzhou, China, November 2012.

[54] S. Paul and P. K. Roy, "Optimal design of power system stabilizer using oppositional gravitational search algorithm," in *Proceedings of the 2014 1st International Conference on Non Conventional Energy (ICONCE 2014)*, pp. 282–287, IEEE, West Bengal, India, January 2014.

[55] S. M. A. Bulbul and P. K. Roy, "Quasi-oppositional gravitational search algorithm applied to complex economic load dispatch problem," in *Proceedings of the 2014 1st International Conference on Non Conventional Energy (ICONCE 2014)*, pp. 308–313, IEEE, West Bengal, India, January 2014.

[56] M. Basu, "Quasi-oppositional group search optimization for hydrothermal power system," *International Journal of Electrical Power & Energy Systems*, vol. 81, pp. 324–335, 2016.

[57] M. Basu, "Combined heat and power economic dispatch using opposition-based group search optimization," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 819–829, 2015.

[58] C. Yang, J. K. Zhang, and L. X. Guo, "Investigation on the inversion of the atmospheric duct using the artificial bee colony algorithm based on opposition-based learning," *International Journal of Antennas and Propagation*, vol. 2016, Article ID 2749035, 10 pages, 2016.

[59] R. Rao, "Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19–34, 2016.

[60] Z. Yin, J. Liu, W. Luo, and Z. Lu, "An improved Big Bang-Big Crunch algorithm for structural damage detection," *Structural Engineering and Mechanics*, vol. 68, no. 6, pp. 735–745, 2018.

[61] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution algorithms," in *Proceedings of the 2006 IEEE International Conference on Evolutionary Computation*, pp. 2010–2017, IEEE, Vancouver, BC, Canada, July 2006.

[62] J. J. Liang, B. Y. Qu, and P. N. Suganthan, *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*, Technical Report, Computational Intelligence Laboratory, Zhengzhou University; Nanyang Technological University, Zhengzhou, China, Singapore, 2013.

[63] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[64] E. Nabil, "A modified flower pollination algorithm for global optimization," *Expert Systems with Applications*, vol. 57, pp. 192–203, 2016.