Soft Computing
https://doi.org/10.1007/s00500-022-07726-0

OPTIMIZATION

Artificial locust swarm optimization algorithm

Orhan Kesemen[1] · Eda Özkul[1] · Özge Tezel[1] · Buğra Kaan Tiryaki[1]

Accepted: 2 December 2022

Abstract
This study proposes a new metaheuristic algorithm, which is called Artificial Locust Swarm Optimization (ALSO), inspired by random jumping and plant invasion behavior of locust swarms. Locusts interact in two different ways of searching for food: social and familial. In the familial phase, small locust groups search foods in a local area and the locusts share their information in the social phase. The proposed algorithm is less likely to trap into the local solution than other methods and has high performance in the sensitivity of the global solution. In addition, it is effective not only for the solution of black-box optimization problems but also for the solution of problems with an irregular objective function. The ALSO algorithm is compared with other recent and well-known optimization algorithms on 22 benchmark functions and 3 real engineering design problems. Simulation results prove that the ALSO algorithm is very competitive when compared to the other algorithms. Moreover, it even requires the less runtime and memory space under the same conditions.

Keywords Optimization · Swarm intelligence · Metaheuristic · ALSO

1 Introduction

In recent years, the rapid development of sciences such as machine learning and artificial intelligence has increased the importance of optimization algorithms. Metaheuristic optimization algorithms have been developed to solve iteratively the problems which cannot be solved analytically. The main goal is to estimate the parameter that minimizes/maximizes the desired objective function. Many algorithms have been developed in recent years to solve optimization problems. They focus on the modeling of the evolutionary processes or lifestyles of living creatures in natural life (Meetei 2014; Yang 2014; Hassanien and Emary 2016; Orujpour et al. 2020. These algorithms, which are based on physical, chemical, and biological processes of living creatures in nature, are preferred because they provide simple, fast, and effective results in solving hard problems. Optimization algorithms are applied to many fields such as image processing, data analysis, machine learning, medicine, finance, data mining, and game theory (Martens et al. 2011; Ahmed and Glasgow 2012; Kesemen and Özkul 2018. Fister et al. (2013) classified the optimization algorithms into four categories: swarm intelligence (SI) based, bio-inspired (but not SI-based), physics/chemistry-based, and others.

Many of the algorithms that are inspired by biological systems directly use swarm intelligence. The most common swarm intelligence based algorithms are the ant colony algorithm (Dorigo and Caro 1999), the particle swarm optimization (Eberhart and Kennedy 1995), the artificial bee colony algorithm (Karaboga and Basturk 2007), the firefly algorithm (Yang et al. 2010), the cuckoo search algorithm (Yang and Deb 2009), the bat algorithm (Yang 2010), the grey wolf optimizer (Mirjalili et al. 2014), the ant lion optimizer (Mirjalili 2015), the whale optimization algorithm (Mirjalili and Lewis 2016), the owl search algorithm (Jain et al. 2018), the pathfinder algorithm (Yapici and Cetinkaya 2019), the emperor penguins colony (Harifi et al. 2019), the squirrel search algorithm (Jain et al. 2019)(Jain, Singh, & Rani, 2019), the butterfly

✉ Orhan Kesemen
okesemen@gmail.com

Eda Özkul
eda.ozkul.gs@gmail.com

Özge Tezel
ozge_tzl@hotmail.com

Buğra Kaan Tiryaki
bugrakaantiryaki@gmail.com

[1] Department of Statistics and Computer Sciences, Karadeniz Technical University, 61080 Trabzon, Turkey

Published online: 19 December 2022

optimization algorithm (Arora and Singh 2019), the harris hawks optimization (Heidari et al. 2019), the red deer algorithm (Fathollahi-Fard et al. 2020), the marine predators algorithm (Faramarzi et al. 2020), the nomadic people optimizer (Salih and Alsewari 2020), the battle royale optimization algorithm (Farshi 2021), and the rat swarm optimizer (Dhiman et al. 2021).

The genetic algorithms (Holland 1975) and the flower pollination algorithm (Yang 2012) are the bio-inspired algorithms which are not directly based on swarm intelligence. The genetic algorithm, improved by Holland Holland (1975), is a biological model based on Darwin's evolution theory. Genetic algorithms provide alternative solutions to problems. Each of these solutions is called a chromosome and chromosomes are composed of genes. The cluster of chromosomes is named as the population (Holland 1975; Lim 2014). The flower pollination algorithm developed by Yang (2012) is based on the pollination process of flowering plants.

Some algorithms simulate physical and chemical laws such as electrical charges, gravity, river systems (Hassanien and Emary 2016; Fister et al. 2013). The most known of these algorithms are simulated annealing (Kirkpatrick et al. 1983), harmony search (Geem et al. 2001), spiral optimization (Tamura and Yasuda 2011), and water cycle algorithm (Eskandar et al. 2012).

This paper introduces a metaheuristic algorithm based on the behavior of the locusts. Although there are several studies inspired locusts in the literature, they focus on different behavior of the locusts. The first study based on locust swarms was proposed by Chen (2009) and it depends on Waves of Swarm Particles (WoSP) which is derived from the particle swarm optimization. Topaz et al. (2008) concentrated on the locusts in the gregarious phase and established a mathematical model for this. Then, Topaz et al. (2012a) modeled density-dependent interconversions, social interactions, and movements of the polyphonic locust for the solitarious and gregarious phase. Another study based on locust swarms was carried out by Cuevas et al. (2015), consists of two phases: solitary and social. In the first phase, the locusts explore the search area. In the second phase, the existing solutions are improved according to the neighborhood. Although this model is based on the model developed by Topaz et al. (2008), it assumes that locusts interact with each other during the solitary phase (Cuevas et al. 2015). Saremi et al. (2017) proposed a grasshopper optimization algorithm, which uses the model developed by Topaz et al. (2008), to solve single-objective problems and it depends on the social interactions of the grasshoppers.

In general, optimization algorithms try to achieve a better solution for one or more randomly selected initial solutions with iterative approaches. Metaheuristic algorithms emphasize two concepts in the optimization process: exploration and exploitation. Each developed metaheuristic algorithm focuses on making a trade-off between the exploration and exploitation to increase the performance of the algorithm (Rashedi et al. 2009). The exploration searches the promising regions of the search space to reach the global optimum and avoids trapping into a local optimum. However, the exploitation searches the optimal solution around the promising regions. Most optimization algorithms concentrate on exploration at the beginning of the search process, while they focus on exploitation in the later stages. The algorithm converges faster when there is too little focus on exploration and too much on exploitation. However, in this case the probability of finding the true global optimal may be less. Similarly, concentrating more on the exploration and less on the exploitation may result in slower convergence. Therefore, the balance between exploration and exploitation plays an important role in the performance of the algorithm. In addition, an important criterion for the performance of the optimization algorithm is that computation time, especially, in high-dimensional complex optimization problems.

In recent years, a numerous new metaheuristic algorithm has been developed, and applied to different real-world and complex optimization problems in many fields. Regardless of this, the question occurs whether new techniques are required in this area (Zhang et al. 2017). The metaheuristic algorithms use different methodologies and procedures, and each has its own mechanism for the exploration and exploitation phases. The no-free-lunch (NFL) theorem (Wolpert and Macready 1997) states that there is no optimization algorithm that can be most efficient to solve every optimization problem. According to this theorem, even if a particular algorithm is better than any other algorithms, it can give worse results for some optimization problems. There is no heuristic algorithm yet that can achieve the best solution for all optimization problems. This theory proves that new techniques can overcome complex and unsolved problems. Hence, it is the motivation of this study to develop a new metaheuristic algorithm inspired by the behavior of locust swarms.

The ALSO is a population-based algorithm like other existing swarm-based algorithms and uses a population consisting of candidate solutions to achieve the best solution. Each solution is conceived as a family which is trying to find random new solutions around a main locust.

The remainder of this paper is organized as follows. In Sect. 2, the proposed ALSO algorithm is elaborated. Section 3 contains the experimental results and performance evaluation. In Sect. 4, the performance of the ALSO is tested on three engineering design problems. Finally, Sect. 5 summarizes the conclusions from the study.

2 Artificial locust swarm optimization (ALSO)

In this section, the main inspiration of the proposed algorithm is first presented. Then, the ALSO algorithm is discussed in detail.

2.1 Behavior of locust swarms

The locusts are a phylogenetically heterogeneous group of the family *Acrididae*. Depending on the population density, locusts change the phase from *'solitarious'* to *'gregarious'*. The phase transition is termed *'solitarization'* and *'gregarization'* (Topaz et al. 2012a). Phase transition is related to behavior, color, morphometry, reproductive development, fertility, endocrine physiology, metabolism, molecular biology (Collett et al. 1998; Simpson et al. 1999, 2011; Kang et al. 2004; Ernst et al. 2015). In the *'solitarious'* phase, the population density is low, and locusts avoid each other. In the *'gregarious'*phase, locusts live in swarms. Thus, they can initiate large migrations (Simpson et al. 2011; Simpson and Sword 2008). When phase transition occurs from *'solitarious'* to *'gregarious'* phase, locusts form enormous groups of countless individuals spanning hundreds of kilometers. The most obvious effects of phase transition are changes in morphological appearance, including body size and color. While the transition from the *'solitarious'* phase to the *'gregarious'* phase occurs within hours, changes in color and morphological features occur over a longer period of time (Simpson et al. 2011; Ernst et al. 2015).

Temperature and wind affect the population density and swarm formation. With the increase of temperature, the plants grow, and the locusts can supply their nutritional needs more easily. The wind allows the locust to fly, and the direction of the wind and movement of the swarm are the same. Unless locusts use their wings while in the air, they have no control over their trajectory of movement (Hoyle 1958). In addition, temperature and wind affect the speed of the swarm. At the end of the warm season, the female locusts lay eggs on their location. The eggs develop in moist soil and hatch in 10–20 days depending on the temperature. The baby locusts, referred as nymphs, have no wings. When a nymph becomes an adult, its wings are formed. Thus, the speed of the swarm increases (Simpson and Sword 2008). The migratory locusts may travel about 100 km per day. In addition, locusts invade the plants in their locations, during migration. Therefore, they can cause significant damage to agricultural areas (Inglis et al. 2007).

Locusts have a great jumping ability and leap randomly in their area. There are species of locusts that can jump about 10 times their body length vertically and about 20 times horizontally (Simpson et al. 2011; Hoyle 1958; Scott 2005; Topaz et al. 2012b). This jump is too fast to be followed by the eye (Hoyle 1958).

Locusts do not have features such as observing or sniffing around their region. Therefore, they search for food by jumping in random directions to find well food sources. The jumping distance depends on the quality of the food sources. When the food source is good, the jumping distance is less. Otherwise, it is large. Locust swarms move around by flying and search the best food sources. During the flight, they see all the regions and have information about the quality of their food search. Hence, the proposed algorithm is developed by the inspiration of the foraging behavior of locust swarms.

Locusts are far from intelligent and social behavior approach compared to other swarming insect species. They are one of the invasive species with random behavior. The proposed algorithm is based on the jumping (exploration) and invasion (exploitation) behavior of locust swarms. In the meantime, the locust's jumping and invasion are modeled with the jumping distance and precision function, respectively. Thus, the exploration and exploitation ability of the proposed method is improved and a proper balance between these two concepts is provided.

2.2 ALSO algorithm

Artificial locust swarm is a hierarchical swarm that consists of small groups (families). Each family includes a main locust. The main locust performs trials with its offspring in the search space to figure out the best solution around the area and retains its position (Fig. 1). The best locust in the family including the main locust survives and passes the social phase. Thus, it joins the swarm and the other family members die. The locusts in the swarm converge to the global solution by continuing the same process from their position in the next generation.

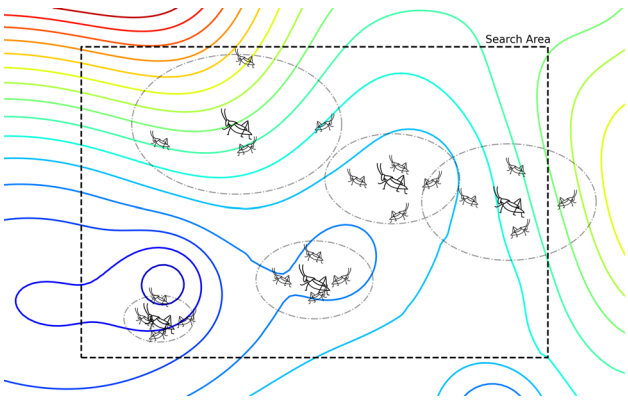

Fig. 1 Representative illustration of the behavior of the family of locusts in the search area

2.2.1 Producing initial position of locusts

The ALSO algorithm starts with randomly producing the initial position of locusts that correspond to the solutions in the search space. Initial positions are generated randomly in the range of the search space as follows.

$$x_{ij} = x_j^{\min} + \text{rand}(0,1)\left(x_j^{\max} - x_j^{\max}\right) \tag{1}$$

where $i = 1, \ldots, NF$, $j = 1, \ldots, D$, NF is the number of families, D is the dimension of the objective function, x_{ij} is the position of ith locust in the jth dimension.$x_j^{\min}$ and $x_j^{\max}$ are the lower and upper bounds of the jth dimension in problem space, respectively.

2.2.2 Determining maximum displacement distance

The ALSO algorithm is based on locust jumping in random directions and distances while locusts search foods in local areas. In the familial phase, locusts prefer jumping instead of flying. How far a locust jumps depends on the quality of food in that locust's region. The quality of an area corresponds to fitness value and, thus maximum displacement distance decreases in areas where there is a rich quality of food and increases in areas where it is low. In other words, the fitness value is inversely proportional to the maximum displacement distance. The inverse fitness function for the minimization problems is represented as follows.

$$u_i = \frac{f\left(\mathbf{x}_i^{(t)}\right) - \min f\left(\mathbf{x}^{(t)}\right)}{\max f\left(\mathbf{x}^{(t)}\right) - \min f\left(\mathbf{x}^{(t)}\right)}, i = 1, 2, \ldots, NF \tag{2}$$

where, $u_i \in [0,1]$ is the inverse fitness value of the i^{th} area, $\mathbf{x}_i^{(t)}$ is i^{th} solution vector in t^{th} iteration, $\mathbf{x}^{(t)}$ all solution vectors in t^{th} iteration, $f(\mathbf{x})$ is the objective function of the problem, $f\left(\mathbf{x}_i^{(t)}\right)$ is the objective value of the i^{th} area in t^{th} iteration, $\min f\left(\mathbf{x}^{(t)}\right)$ is the minimum objective value in t^{th} iteration, $\max f\left(\mathbf{x}^{(t)}\right)$ is the maximum objective value in t^{th} iteration and NF is the number of families. The inverse fitness function for the maximization problems is $1 - u_i$.

Maximum displacement distance is calculated by Eq. (3).

$$\mathbf{r_i} = \mathbf{r}_0(u_i + Precision(t, T, \epsilon)), i = 1, 2, \ldots, NF \tag{3}$$

where $\mathbf{r_i}$ is the maximum displacement distance, $\mathbf{r}_0$ refers to the quarter width of the search space for each dimension and is determined as in Eq. (4).

$$\mathbf{r}_0 = \frac{(\mathbf{x_{max}} - \mathbf{x_{min}})}{4}, \mathbf{r}_0, \mathbf{x_{max}}, \mathbf{x_{min}} \in \mathbb{R}^D \tag{4}$$

Here, $\mathbf{x_{max}}$ and $\mathbf{x_{min}}$ are the upper and lower bound vector of the search space. In the Eq. (3), it is preferred that two piecewise precision function as in Eq. (5).

$$Precision(t, T, \epsilon, \lambda = 0.5) = \begin{cases} 1 & t \leq \lambda T \\ \epsilon^{\left(\frac{t-\lambda T}{T-\lambda T}\right)} & t > \lambda T \end{cases}, (t = 1, 2, \ldots, T) \tag{5}$$

Here, T is the maximum iteration number, t is the current iteration, ϵ is the expected minimum precision, λ ranges in $(0,1)$ and determines the inflection point (λT) of the precision function. While the precision function is constant to λT, it decreases exponentially after λT (Fig. 2). In this study, λT states the half of the maximum iteration number because of the case $\lambda = 0.5$. Furthermore, λT varies in respect to λ.

Until the inflection point of the precision function, the global solution is searched. Afterwards, the sensitivity of the global solution is improved. In the metaheuristic optimization algorithm, the sensitivity of the solution depends on the interaction of the agents. Therefore, it cannot affect the sensitivity of the solution. However, it can be reached the desired sensitivity by using ε owing to the precision function.

2.2.3 Producing new offspring of the main locust

In the ALSO algorithm, the search is performed by the main locust and its offspring. In each iteration, the main locust in each family leaves the swarm and moves to their new location and lays eggs there. Thus, new offspring are produced in different areas of the search space. The positions in which the main locust will lay eggs are calculated using the maximum displacement distance and the current position of the main locust. Therefore, the position of an offspring is represented in Eq. (6).

$$v_{k,J} = x_{i,J} + rand(-\mathbf{r}_i, \mathbf{r}_i), i = 1, 2, \ldots, NF \tag{6}$$

where $v_{k,J}$ and $x_{i,J}$ are the position of the k^{th} offspring and i^{th} main locust, respectively. J is random dimension of problem space.

Offspring hatch from eggs in new position, and thus new locust families are generated in new local areas. In new area, fitness values of each new offspring in all families are evaluated according to objective function. Then, the fitness values of each area are calculated by using Eq. (2) and the position of the best area is updated. When the maximum iteration number is reached, the algorithm outputs the best solution and its fitness value.

The pseudo-code of the ALSO algorithm is explained in Algorithm 1. Some fundamental characteristics about the proposed ALSO algorithm is given below.

- The precision function in ALSO algorithm provides the proper balance between the exploitation and exploration. In addition, the global solution can be reached with the desired sensitivity in virtue of this function.
- The ALSO algorithm saves the obtained best solution during the course iterations.
- How far the locusts will jump is determined by the quality of each area.

In Algorithm 1, the function `rand(a,b)` generates a random number in the range of [a, b) from the uniform distribution, the function [a, b] from the set of integers.

Algorithm 1 Pseudo Code of Artificial Locust Swarm Optimization

```
function also(f, N, D, T, eps,xmin,xmax)
" f(.): objective function
" N   : Number of total locusts
" D   : Dimension of objective function
" T   : Number of Maximum Iteration
" eps : The value of expected minimum precision
" xmin,xmax: lower and upper bound of search space
" precision(.)   : precision function

" initialize d-dimensional all locust
FS = D / 2       " initialize each family size
NF = N / FS      " initialize number of families

r0_j = (xmax_j − xmin_j) / 4 → (j = 1,2, … , D)
for family_i in (1 to NF) do
    x_{i,j} = rand(xmin_j , xmax_j) → (j = 1,2, … , D)
    f_i = f(x_{i,:})

" start iterations
for t in (1 to T) do

    " social phase
    for family_i in (1 to NF) do
        " update the inverse fittness values
        u_i = (f_i − min(f)) / (max(f) − min(f))

    " familial phase
    for family_i in (1 to NF) do
        xNext_j = x_{i,j} → (j = 1,2, … , D)      " handled locust

        " trial new positions
        for trial_k in (1 to FS) do
            v_j = x_{i,j} → (j = 1,2, … , D)
            J = irand(1, D)      " selection of dimension

            " update maximum displacement distances
            r = r0_J * (u_i + Precision(t, T, eps))

            " update of candidate solution
            v_J = x_{i,J} + rand(−r, r)

            " check the bounds of j-th dimension bounds
            if is not xmin_J < v_J < xmax_J
                v_J = rand(xmin_J, xmax_J)

            " selection of candidate solution
            if f(v) < f_i
                xNext_j = v_j → (j = 1,2, … , D)
                f_i = f(v)
        " update i-th better positions
        x_{i,j} = xNext_j → (j = 1,2, … , D)
return x
```

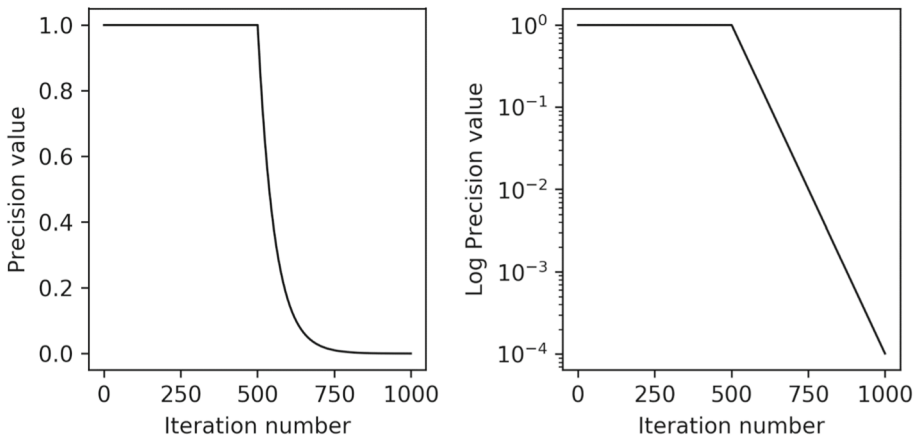

Fig. 2 The precision functions

2.3 Computational complexity of the ALSO algorithm

The computational complexity of an algorithm is an important measure to evaluate the amount of memory needed and runtime. The computational complexity of the ALSO depends on the number of locusts (N), maximum iteration number (T). Therefore, the overall computational complexity is defined as follows:

$$O_{time}(ALSO) = O(T(O(positionupdate))) \tag{7}$$

$$O_{time}(ALSO) = O(TN). \tag{8}$$

The computational complexity of the ALSO is not affected by the number of variables (D) of the problem, since each locust moves through only one dimension in the familial phase.

Moreover, memory usage is an important factor in solving high-dimensional complex problems. Since the ALSO stores only the position of the best member of the families at each iteration, the space complexity is defined as follows:

$$O_{space}(ALSO) = O\left(\frac{2N}{D} \times D\right) \tag{9}$$

$$O_{space}(ALSO) = O(N) \tag{10}$$

Therefore, it can be pointed out that the ALSO has good performance in both time complexity and space complexity.

3 Experimental results and performance evaluation

To evaluate the performance of the proposed ALSO algorithm, a comprehensive experimental evaluation and comparison have been performed with the most relevant and recent algorithms: PSO (the particle swarm optimization) (Lynn and Suganthan 2015), ABC (the artificial bee colony) (Karaboga and Basturk 2007), CS (the cuckoo search) (Yang and Deb 2009), BAT (the bat-inspired algorithm) (Yang 2010), GWO (the grey wolf optimizer) (Mirjalili et al. 2014), WOA (the whale optimization algorithm) (Mirjalili and Lewis 2016), HHO (the harris hawks optimization) (Heidari et al. 2019), BRO (the battle royale optimization algorithm) (Farshi 2021), and RSO (the rat swarm optimizer) (Dhiman et al. 2021). The controlling parameters for all algorithms were set according to the recommendations in the original papers. Moreover, all algorithms were implemented in MATLAB R2019a, and performed a computer with an Intel Core i5-6400 CPU 2.70 GHz processor, 8 GB RAM was used.

3.1 Performance evaluation of ALSO according to parameters

The relationship between the number of families (NF) and the number of locusts in each family (FS) has been simulated in the proposed algorithm. The number of families and number of locusts in each family were determined as $NF = \{2, 3, 4, 5, 6, 10, 12, 15, 20, 30\}$ and $FS = \{30, 20, 15, 12, 10, 6, 5, 4, 3, 2\}$, respectively. Therefore, it was taken as the total number $N = NF \times FS = 60$ of locusts. The proposed method was run 30 times for the dimension $D = 30$. The obtained results for some functions are shown in Fig. 3.

According to Fig. 3, when the proposed algorithm is analyzed by the number of families, it is observed that the sensitivity of the solution increases when the number of families decreases in some functions. As the number of locusts increases, the sensitivity of the global solution also improves. For the Griewank function, the ALSO has the best result when $FS = 15$. For the other functions, the ALSO gives the best result when $FS = 20$ and.

$FS = 30$. The convergence curves of $FS = 30$, $FS = 20$ and $FS = 15$ have close results especially for Schwefel

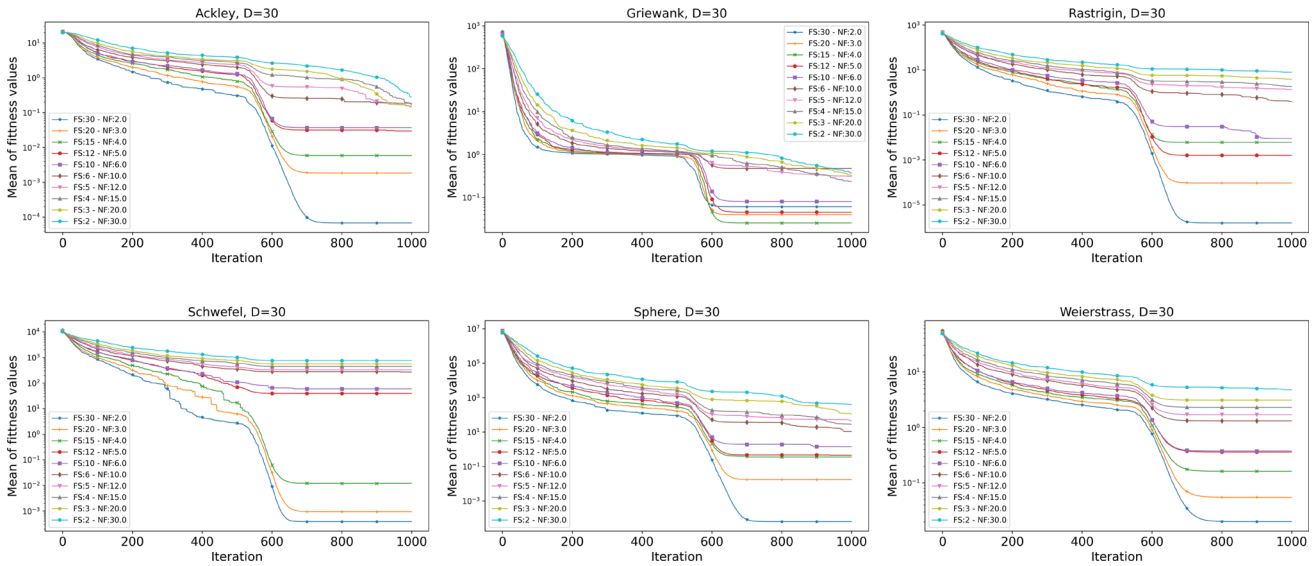

Fig. 3 Evolution of the ALSO algorithm by the number of groups

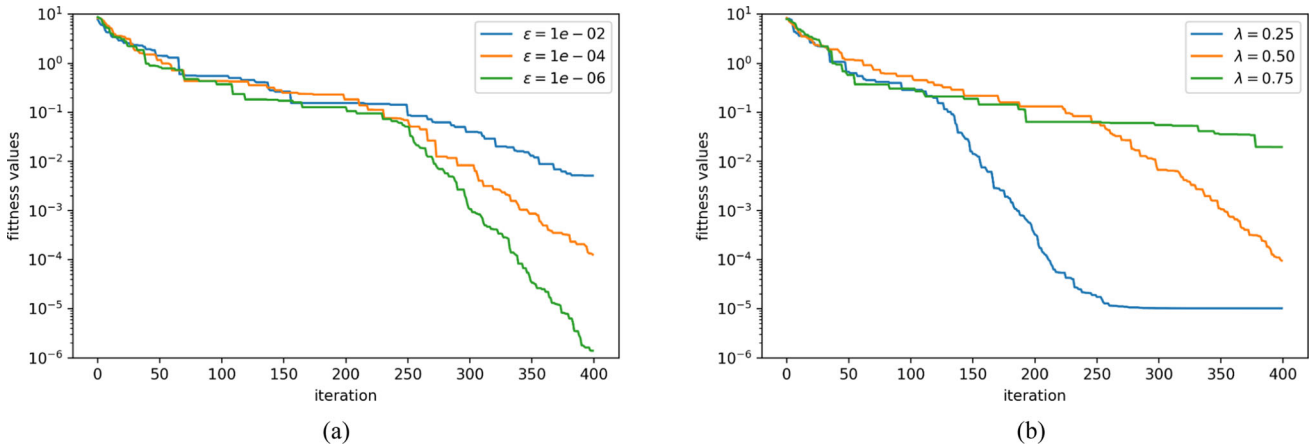

Fig. 4 The best fitness values according to ϵ and λ **a** The best fitness values according to ϵ with $\lambda = 0.5$ **b** The best fitness values according to λ with $\epsilon = 1e-04$

function. Thus, considering the balance between exploration and exploitation, it is concluded that $FS = 15$ is more appropriate for the simulations. Therefore, in this study, the number of locusts in a family (FS) is determined as half of the dimension ($D/2$).

The ALSO algorithm was evaluated on Ackley function to demonstrate the change of the best fitness values according to the parameters ϵ and λ with different values. Figure 4 shows the best fitness values in each iteration.

Figure 4a concludes that half of the total iteration focused on exploration, while the other half focused on exploitation. Therefore, the three convergence curves show similar decreases in the first half of the total iteration ($t < 200$). The difference in ϵ values in the last half of the iteration reveals how much it should improve exploitation.

In Fig. 4b, while the convergence curve of $\lambda = 0.25$ with $\epsilon = 1e-04$ shows exploration in the first quarter of the total iterations ($t < 100$), performs exploitation in the other quarters. In this case, premature converges may occur and, then it may cause with trapping into the local solutions. For the convergence curve of $\lambda = 0.50$, the algorithm focuses on exploration in the first half of the total iterations

($t < 200$) and emphasizes on exploitation in the other half. Moreover, it can be observed from the convergence curve of $\lambda = 0.75$, algorithm concentrations on exploitation during the third quarters ($t > 300$). In this case, converging to the global solution may not be satisfactory.

3.2 Comparison of ALSO with other algorithms

In this stage, the ALSO is tested on 22 benchmark test functions which are grouped as unimodal, multimodal, and composite. The first 6 benchmark functions are unimodal functions. The second 6 functions are the multimodal functions. On the other hand, the last 10 benchmark functions are the composite functions which are handled in CEC 2017 (Awad et al. 2017). Table 1 gives the ranges and optimal values of unimodal functions, and Table 2 lists the ranges and optimal values of multimodal functions, and they are generalized to be defined in the dimensions 30, 50, 100. Table 3 tabulates the ranges and transform parameters of composite functions, which are available in the CEC 2017 technical report (Awad et al. 2017), and they are generalized to be defined in the dimensions 30, 50, 100. Figures 5, 6, 7 illustrate the 2D versions of the benchmark functions used.

In order to demonstrate the performance of the proposed method, it was benchmarked on unimodal, multimodal and composite benchmark functions and was compared with well-known algorithms, and all runs were conducted under the same conditions. The maximum iteration number (T) was set to 3000 for each unimodal, multimodal, and composite benchmark function with 30, 50 and 100 dimensions and all algorithms were used 150 search agents.

The ALSO and all compared algorithms were run 30 times and the mean and standard deviation of the best values according to optimal value of the functions are listed in Tables 4, 5, 6 for unimodal benchmark functions,

in Tables 7, 8, 10 for multimodal benchmark functions, and in Tables 10, 11, 12 for composite benchmark functions. The standard deviation of 30 runs was also calculated. The computational time (sec), mean and standard deviation (std) are three main criteria that are often use the demonstrate the performance of optimization algorithms. Moreover, standard deviation values are to figure out the stability of the algorithms over the 30 independent runs and computational time indicates computational efforts of the algorithms. In the comparison of the results, it will be meaningless to use values smaller than the machine epsilon ($2.22E-16$), so those less than it, are equal to the machine epsilon value.

As Table 4 shows, the ALSO, PSO, GWO, WOA, HHO, BRO, and RSO give the best results for F1. F2 is obtained by shifting F1 to the right 1 unit. The ALSO, PSO, and BRO provide the best results for F2. Although F2 is the shifted form of F1, the GWO, WOA, HHO, and RSO are not as effective as F1. The ALSO, PSO, ABC, and BRO attain good performance for F3. The ALSO outperforms all the other algorithms for F4 and F5. The ALSO and BRO outperform all the other algorithms for F6. Moreover, it can be observed that in most cases ALSO provides a better standard deviation when compared to the other algorithms. Based on average running time on 6 unimodal benchmark functions, the ALSO performs faster than the other optimizers.

Table 5 reveals that the ALSO, PSO, GWO, WOA, HHO, and RSO give the best results for F1. The ALSO and PSO provide the best results for F2. Although F2 is the shifted form of F1, the GWO, WOA, HHO, and RSO are not as effective as F1. The ALSO, PSO, and BRO have

Table 1 Unimodal benchmark functions

F	Name	Function	Range	Optimal Value
F1	*Sphere*	$f_1(\vec{x}) = \sum_{j=1}^{D} x_j^2$	$-100 \le x_j \le 100$	$f_1(\vec{0}) = 0$
F2	*Sphere M1*	$f_2(\vec{x}) = \sum_{j=1}^{D} (x_j - 1)^2$	$-100 \le x_j \le 100$	$f_2(\vec{1}) = 0$
F3	*Exponential M1*	$f_3(\vec{x}) = 1 - \exp\left(-0.5\sum_{j=1}^{D} (x_j - 1)^2\right)$	$0 \le x_j \le 2$	$f_3(\vec{1}) = 0$
F4	*Sum of Different Powers M1*	$f_4(x) = \sum_{j=1}^{D} \lvert x_j - 1\rvert^{j+1}$	$-10 \le x_j \le 10$	$f_4(\vec{1}) = 0$
F5	*Schwefel 2.22 M1*	$f_5(\vec{x}) = \sum_{j=1}^{D} \lvert x_j - 1\rvert + \prod_{j=1}^{D} \lvert x_j - 1\rvert$	$-100 \le x_j \le 100$	$f_5(\vec{1}) = 0$
F6	*Sum Squares M1*	$f_6(x) = \sum_{j=1}^{D} j(x_j - 1)^2$	$-10 \le x_j \le 10$	$f_6(\vec{1}) = 0$

Table 2 Multimodal benchmark functions

F	Name	Function	Range	Optimal Value
F7	*Ackley*	$f_7(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{j=1}^{D} x_j^2}\right) - \exp\left(\frac{1}{D}\sum_{j=1}^{D}\cos(2\pi x_j)\right) + 20 + \exp(1)$	$-32.768 \le x_j \le 32.768$	$f_7\left(\vec{0}\right) = 0$
F8	*Rastrigin*	$f_8(\vec{x}) = \sum_{j=1}^{D}\left(x_j^2 - 10\cos(2\pi x_j) + 10\right)$	$-5.12 \le x_j \le 5.12$	$f_9\left(\vec{0}\right) = 0$
F9	*Rosenbrock*	$f_9(\vec{x}) = \sum_{j=1}^{D-1} 100\left(x_{j+1} - x_j^2\right)^2 + (1 - x_j)^2$	$-5 \le x_j \le 5$	$f_{10}\left(\vec{1}\right) = 0$
F10	*Griewank*	$f_{10}(\vec{x}) = 1 + \sum_{j=1}^{D} \frac{x_j^2}{4000} - \prod_{j=1}^{D}\cos\left(\frac{x_j}{\sqrt{j}}\right)$	$-600 \le x_j \le 600$	$f_8\left(\vec{0}\right) = 0$
F11	*Schwefel*	$f_{11}(\vec{x}) = 418.9829 \cdot D - \sum_{j=1}^{D} x_j \sin\left(\sqrt{\lvert x_j \rvert}\right)$	$-500 \le x_j \le 500$	$f_{11}\left(\overrightarrow{420.9687}\right) \approx 0$
F12	*Weierstrass*	$f_{12}(\mathbf{x}) = \sum_{j=1}^{D}\left(\sum_{k=0}^{20}\frac{1}{2^k}\cos\left(2\pi 3^k\left(x_j + \frac{1}{2}\right)\right)\right) - D\sum_{k=0}^{20}\frac{1}{2^k}\cos\left(\pi 3^k\right)$	$-5 \le x_j \le 5$	$f_{12}\left(\vec{0}\right) = 0$

good performance for F3. The ALSO outperforms all the other algorithms for F4, F5, and F6. In addition, ALSO has reached to the global optimum in most problems with accuracy close to the high-performance optimizers. The experiments indicate that ALSO provides faster results overall, however.

As per result in Table 6, the GWO, WOA, HHO, and RSO give the best results for F1. However, the ALSO provides good performance for F1. The ALSO and HHO present better results for F2. Although F2 is the shifted form of F1, the GWO, WOA, and RSO are not as effective as F1. The ALSO and PSO perform better than the other compared optimizers for F3. The ALSO outperforms all the other algorithms for F4 and F5. The ALSO and HHO present best performance for F6. Based on the standard deviation on 6 unimodal benchmark functions for 100 dimensions, the ALSO performs accurately when compared to the other optimizers. Moreover, according to running time, the ALSO is faster than the other compared algorithms.

As can be seen from Tables 4–6, the ALSO performs very competitive results for unimodal benchmark functions on the dimensions 30, 50, 100. In addition to this, the GWO, WOA, HHO, and RSO work well when the solution is at point 0. However, as the solution moves away from point 0, they move away from the optimal value. Consequently, it is concluded that they are not reliable.

The best fitness and runtime values for the unimodal benchmark function for 30, 50 and 100 dimensions are also shown in Fig. 8. In order to better demonstrate the changes in the graph, the mean of best fitness values greater than 10^4 are set to 10^4.

Figure 8 demonstrates that the ALSO algorithm provides best superior results in terms of the best fitness value for all unimodal functions for all dimensions. As indicated in the results, the ALSO can reach the global solution if it has the least time to search.

Figure 9 depicts the convergence curves of ALSO and other compared algorithms on 3000 iterations for all dimensions and illustrates that the mean of the best fitness values of unimodal benchmark functions in each iteration for all compared algorithms. It is shown that ALSO is very competitive over other metaheuristic techniques. It is clearly seen that the convergence curve of the ALSO algorithm is a piecewise function and performs in accordance with the precision function. The first part is half of the maximum iterations because of the case $\lambda = 0.5$ and it can be adapted by changing λ. In addition, the first part

Table 3 Composite benchmark functions

F	Name	Function detail	Range	Bias
F13	Composition function 1	*CEC2017 F21*	$-100 \leq x_j \leq 100$	2100
F14	Composition function 2	*CEC2017 F22*	$-100 \leq x_j \leq 100$	2200
F15	Composition function 3	*CEC2017 F23*	$-100 \leq x_j \leq 100$	*2300*
F16	Composition function 4	*CEC2017 F24*	$-100 \leq x_j \leq 100$	*2400*
F17	Composition function 5	*CEC2017 F25*	$-100 \leq x_j \leq 100$	*2500*
F18	Composition function 6	*CEC2017 F26*	$-100 \leq x_j \leq 100$	2600
F19	Composition function 7	*CEC2017 F27*	$-100 \leq x_j \leq 100$	2700
F20	Composition function 8	*CEC2017 F28*	$-100 \leq x_j \leq 100$	2800
F21	Composition function 9	*CEC2017 F29*	$-100 \leq x_j \leq 100$	2900
F22	Composition function 10	*CEC2017 F30*	$-100 \leq x_j \leq 100$	3000

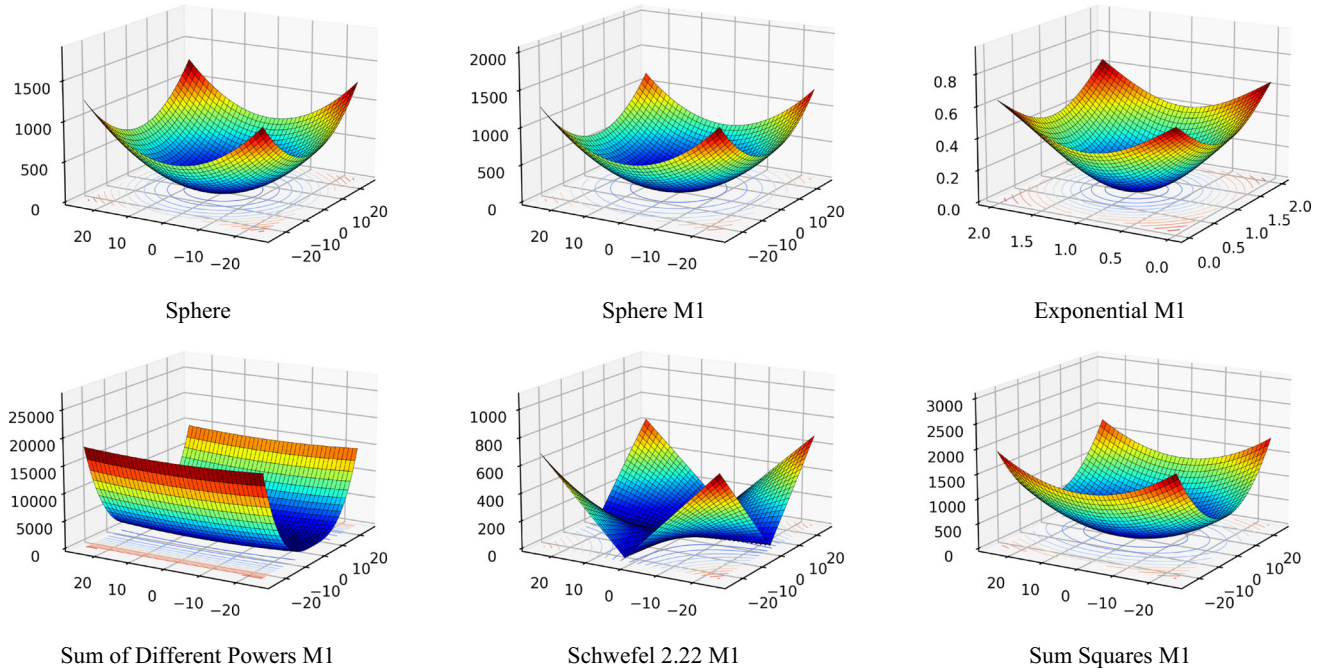

Fig. 5 2-D versions of the unimodal benchmark functions

carries out the exploration, while the second part encourages exploitation. The ALSO algorithm shows a fast converge on the functions F4 and F5 for all dimensions. Moreover, unlike other functions, the convergence speed of the ALSO increases for the second half of the convergence curves. Therefore, it can be said that the ALSO efficiently balances exploration and exploitation to approximate the global optimum.

Table 7 reports that the HHO gives the best result for F7, but the ALSO has remarkable solution. In addition, the ALSO, GWO, WOA, HHO, and RSO provide best results for F8. The HHO produces best result for F9. The GWO, HHO, and RSO perform better results than the other compared optimizers for F10. The ALSO and HHO outperforms all algorithms for F11. The ALSO, PSO, CS, BAT, WOA, and HHO acquire better results than the others for F12. Moreover, ALSO has reached to the global optimum in most problems with accuracy close to the high-performance optimizers. According to running time, the ALSO is faster than the other compared algorithms.

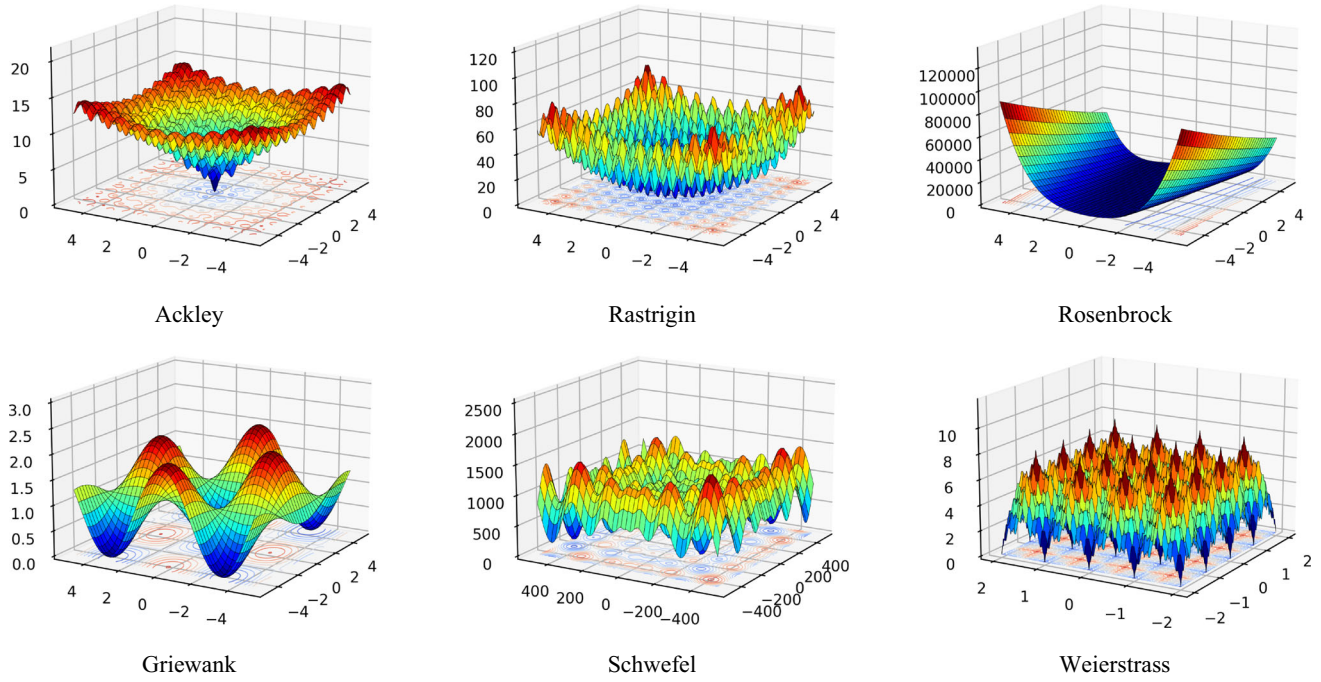

Fig. 6 2-D versions of the multimodal benchmark functions

Table 8 shows that the ALSO, GWO, WOA, HHO, and RSO produce better results than the others for F7 and F8. The HHO gives the best result for F9. The HHO and RSO outperform all compared algorithms for F10. The ALSO produces the best result for F11. The PSO, CS, BAT, WOA, and HHO provide best results for F12. Moreover, it can be observed that in most cases ALSO provides a better standard deviation when compared to the other algorithms. The experiments indicate that ALSO provides faster results overall, however.

Table 9 reveals that the GWO, WOA, HHO, and RSO produce the best results for F7, F8, and F10. The HHO outperforms the other algorithms for F9. The ALSO and HHO perform good performance on F11. The PSO, CS, BAT, WOA, and HHO attain best results for F12. Based on the standard deviation on 6 multimodal benchmark functions for 100 dimensions, the ALSO performs accurately when compared to the other optimizers. Moreover, according to running time, the ALSO is faster than the other compared algorithms.

Figure 10 illustrates that the ALSO algorithm produces best superior results in terms of the best fitness value for all multimodal benchmark functions for all dimensions. In order to better demonstrate the changes in the graph, the mean of best fitness values greater than 10^4 are set to 10^4. Moreover, based on average runtime, the ALSO is faster than the other optimizers.

Figure 11 shows the convergence curves of ALSO and other compared algorithms on 3000 iterations for all dimensions. The results indicate that ALSO is very competitive over the compared optimizers. The convergence speed of the ALSO increases for the second half of the convergence curves. This behavior is due to precision function. Therefore, the ALSO smoothly balances exploration and exploitation, initially underlining local optima avoidance and then convergence.

As Table 10 shows, the ALSO is very competitive as compared to other competitor algorithms for F13, F14, F16, F17, and F19. Moreover, the ALSO maintains better results for functions F15, F18, F20, F21, and F22. In addition, based on standard deviation, the ALSO performs accurately when compared to the other optimizers. According to running time, the results specify that the ALSO has reached the global solution in less time.

Inspecting the results in Table 11, it shows that the ALSO is able to obtain competitive results in many functions and it outperforms other algorithm for F14, F18, F21, and F22. Meantime, it can be observed that in most cases ALSO produces a better standard deviation when compared

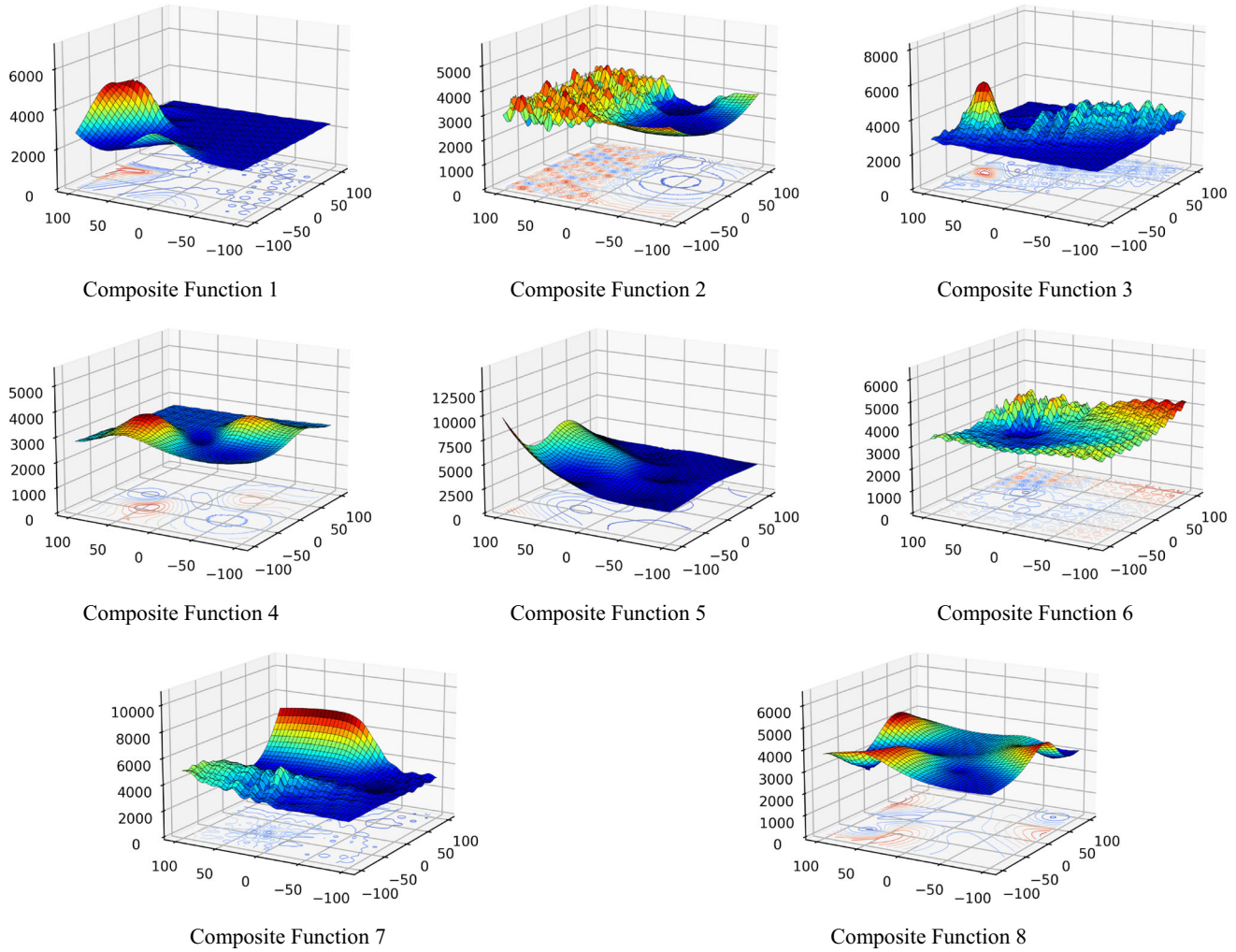

Fig. 7 2-D versions of the composite benchmark functions for F13-F20

to the other algorithms. The experiments indicate that ALSO provides faster results overall, however.

The results in Table 12 specify that the ALSO is able to obtain competitive results for F13 and F18 and it outperforms the compared algorithms for F14, F15, F16, F17, F19, F20, F21, and F22. On the other hand, it can be observed that in most cases ALSO provides a better standard deviation when compared to the other algorithms. Based on average running time on composite benchmark functions, the ALSO performs faster than the other optimizers.

The best fitness and runtime values for the composite benchmark function for 30, 50 and 100 dimensions are also shown in Fig. 12. In order to better demonstrate the changes in the graph, the mean of best fitness values greater than 2×10^4 are set to 2×10^4. It is indicated that the ALSO attains satisfactory results in terms of the best fitness value for all composite functions. Furthermore, the ALSO produces better results according to the runtime.

The convergence curves of ALSO and other compared algorithms on 3000 iterations for the dimensions 30, 50 100 are demonstrated in Figs. 13, 14, 15, respectively. The results reveal that ALSO is very competitive over other metaheuristic techniques. The composite functions are even more challenging than the other benchmark functions. Therefore, they require a proper balance between exploration and exploitation. Thus, it can be indicated that the ALSO is able to balance exploration and exploitation properly for solving such challenging problems. In the meantime, the ALSO algorithm shows a fast converge on

Table 4 Results of unimodal benchmark functions for D = 30

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F1										
Mean	**2.22E-16**	**2.22E-16**	3.93E-15	1.12E-11	3.25E + 01	**2.22E-16**	**2.22E-16**	**2.22E-16**	**2.22E-16**	**2.22E-16**
Std	2.47E-32	2.47E-32	5.53E-15	3.53E-12	9.75E + 01	2.47E-32	2.47E-32	2.47E-32	2.47E-32	2.47E-32
Time(sec)	1.16	16.50	12.02	8.51	3.38	3.96	2.15	7.03	13.47	2.28
F2										
Mean	**2.22E-16**	**2.22E-16**	3.18E-15	1.01E-11	4.22E + 01	2.97E-01	4.11E-06	2.30E-07	**2.22E-16**	2.47E + 01
Std	2.47E-32	2.47E-32	3.34E-15	3.60E-12	1.40E + 02	5.23E-01	1.61E-06	3.10E-07	2.47E-32	1.11E + 00
Time(sec)	1.16	16.36	12.02	8.51	3.37	3.96	1.99	6.87	13.47	2.17
F3										
Mean	**5.74E-16**	**2.22E-16**	**2.22E-16**	6.96E-16	4.68E-06	1.25E-06	2.18E-06	9.30E-07	**2.83E-16**	9.41E-01
Std	1.38E-16	8.01E-21	2.47E-32	1.95E-16	5.63E-07	2.73E-07	1.17E-06	1.34E-06	5.88E-17	1.61E-02
Time(sec)	1.18	16.29	12.14	8.53	3.39	3.98	2.01	7.04	13.49	2.24
F4										
Mean	**2.80E-14**	1.05E + 03	1.26E-01	1.00E + 10	2.46E-04	3.34E-02	6.18E-04	1.23E-05	3.74E-08	2.38E + 01
Std	2.13E-13	7.55E + 03	1.52E-01	0.00E + 00	3.77E-05	1.80E-01	3.68E-04	2.61E-05	2.06E-08	1.29E + 00
Time(sec)	1.68	16.87	12.78	9.84	3.92	4.50	2.53	8.21	14.02	2.68
F5										
Mean	**9.24E-15**	7.66E + 01	9.50E + 01	1.00E + 10	3.11E + 27	9.13E-01	3.04E-02	3.71E-03	5.40E + 02	2.52E + 01
Std	1.68E-15	8.01E + 01	7.48E + 01	0.00E + 00	2.30E + 28	9.12E-01	2.21E-02	2.81E-03	1.24E + 02	7.39E-01
Time(sec)	1.16	16.40	12.18	8.82	3.33	3.96	1.99	6.86	13.45	2.18
F6										
Mean	**2.22E-16**	8.77E + 00	7.54E-16	1.17E-12	1.55E-04	8.49E + 00	2.17E-03	1.72E-06	**2.22E-16**	3.61E + 02
Std	2.47E-32	3.05E + 01	1.75E-15	4.96E-13	3.15E-05	7.33E + 00	2.32E-03	3.20E-06	2.47E-32	1.54E + 01
Time(sec)	1.17	16.35	12.03	8.52	3.36	3.96	1.99	6.93	13.49	2.17

Table 5 Results of unimodal benchmark functions for D = 50

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F1										
Mean	**2.22E-16**	**2.22E-16**	1.30E-01	1.82E-06	8.76E + 02	**2.22E-16**	**2.22E-16**	**2.22E-16**	6.58E-14	**2.22E-16**
Std	2.47E-32	2.47E-32	3.41E-02	4.14E-07	6.68E + 02	2.47E-32	2.47E-32	2.47E-32	1.28E-13	2.47E-32
Time(sec)	1.18	16.84	12.07	10.38	3.67	5.55	2.43	8.46	17.42	2.73
F2										
Mean	**2.22E-16**	**2.22E-16**	1.32E-01	1.67E-06	8.04E + 02	3.08E + 00	8.02E-05	5.88E-07	7.52E-14	4.54E + 01
Std	2.47E-32	2.47E-32	3.89E-02	3.87E-07	7.26E + 02	1.48E + 00	2.77E-05	8.64E-07	2.60E-13	1.21E + 00
Time(sec)	1.17	16.83	12.07	10.39	3.65	5.54	2.22	8.16	17.40	2.56
F3										
Mean	**1.13E-15**	**2.46E-16**	1.01E-05	8.41E-11	1.53E-05	6.27E-02	1.62E-05	1.47E-06	**8.05E-16**	9.95E-01
Std	1.80E-16	4.57E-17	3.54E-06	2.02E-11	1.32E-06	1.52E-01	7.53E-06	2.67E-06	1.59E-16	3.31E-03
Time(sec)	1.19	16.70	12.15	10.41	3.68	5.56	2.24	8.36	17.49	2.65
F4										
Mean	**2.22E-16**	3.16E + 13	8.14E + 11	1.00E + 10	3.42E-04	1.57E + 00	6.45E-04	7.06E-06	4.27E-07	4.49E + 01
Std	2.47E-32	2.37E + 14	1.47E + 12	0.00E + 00	5.96E-05	1.04E + 00	3.26E-04	1.18E-05	1.84E-07	1.03E + 00
Time(sec)	2.01	17.60	13.55	12.43	4.53	6.40	3.07	10.27	18.30	3.38
F5										
Mean	**2.12E-14**	2.65E + 02	3.97E + 28	1.00E + 10	1.36E + 49	5.31E + 00	2.49E-01	6.17E-03	1.01E + 03	4.56E + 01
Std	2.55E-15	1.21E + 02	1.68E + 29	0.00E + 00	1.02E + 50	1.81E + 00	1.39E-01	3.95E-03	1.34E + 02	1.04E + 00
Time(sec)	1.17	16.91	12.58	10.79	3.61	5.53	2.22	8.16	17.52	2.56
F6										
Mean	**2.22E-16**	1.02E + 02	1.97E-02	4.04E-07	1.33E-03	7.42E + 01	4.74E-02	5.24E-06	1.10E-07	1.12E + 03
Std	2.47E-32	2.34E + 02	6.06E-03	1.14E-07	2.77E-04	3.28E + 01	5.95E-02	7.43E-06	6.61E-07	2.86E + 01
Time(sec)	1.18	16.79	12.10	10.40	3.64	5.55	2.22	8.24	17.43	2.57

Table 6 Results of unimodal benchmark functions for D = 100

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F1										
Mean	6.73E-06	5.69E-06	3.59E + 04	4.38E-02	6.17E + 03	**2.22E-16**	**2.22E-16**	**2.22E-16**	3.61E + 03	**2.22E-16**
Std	6.78E-06	1.77E-05	2.51E + 03	9.93E-03	3.01E + 03	2.47E-32	2.47E-32	2.47E-32	7.91E + 02	2.47E-32
Time(sec)	1.24	17.66	12.95	15.13	4.30	9.55	3.22	11.94	30.64	3.91
F2										
Mean	**6.47E-06**	3.27E + 02	3.49E + 04	4.34E-02	6.31E + 03	2.16E + 01	2.88E-03	**1.35E-06**	3.45E + 03	9.76E + 01
Std	5.59E-06	1.76E + 03	2.75E + 03	1.13E-02	2.73E + 03	2.83E + 00	5.17E-04	2.66E-06	7.06E + 02	5.49E + 00
Time(sec)	1.23	17.63	12.93	15.12	4.28	9.51	2.91	11.41	30.70	3.56
F3										
Mean	**4.01E-10**	**3.10E-10**	8.37E-01	2.24E-06	7.09E-05	6.82E-01	6.67E-03	6.85E-06	1.57E-01	1.00E + 00
Std	3.08E-10	1.26E-09	2.06E-02	4.96E-07	4.61E-06	2.14E-01	5.04E-02	1.35E-05	2.64E-02	3.45E-06
Time(sec)	1.25	17.66	13.02	15.15	4.29	9.52	2.94	11.71	30.66	3.69
F4										
Mean	**2.22E-16**	9.64E + 44	2.92E + 52	1.00E + 10	5.30E-04	1.94E + 01	7.81E-04	8.54E-06	7.04E + 18	9.60E + 01
Std	2.47E-32	7.33E + 45	9.84E + 52	0.00E + 00	1.05E-04	3.38E + 00	3.76E-04	1.46E-05	3.48E + 19	2.50E + 00
Time(sec)	2.84	19.59	14.79	18.86	5.90	11.10	4.53	15.48	32.45	5.11
F5										
Mean	**3.38E-04**	7.30E + 02	1.75E + 98	1.00E + 10	6.78E + 105	2.81E + 01	1.70E + 00	1.32E-02	2.24E + 03	9.57E + 01
Std	2.73E-04	2.47E + 02	1.20E + 99	0.00E + 00	5.20E + 106	3.48E + 00	4.79E-01	8.83E-03	2.40E + 02	3.04E + 00
Time(sec)	1.24	17.89	13.07	15.70	4.21	9.50	2.91	11.47	30.64	3.54
F6										
Mean	**2.83E-05**	1.39E + 03	9.48E + 03	2.48E-02	3.03E-02	9.98E + 02	6.56E-01	**1.76E-05**	1.65E + 03	4.82E + 03
Std	2.19E-05	1.29E + 03	7.93E + 02	5.62E-03	8.44E-03	1.64E + 02	3.77E-01	3.27E-05	2.91E + 02	2.12E + 02
Time(sec)	1.26	17.63	12.96	15.15	4.27	9.51	2.92	11.55	30.67	3.55

Table 7 Results for the multimodal benchmark functions for D = 30

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F7										
Mean	5.23E-14	1.04E-14	1.21E-07	5.39E-01	1.25E + 01	6.69E-15	2.47E-15	**2.22E-16**	1.16E-13	2.16E-15
Std	1.23E-14	3.57E-15	1.06E-07	3.50E-01	9.11E-01	1.14E-15	2.30E-15	2.47E-32	6.40E-14	1.64E-15
Time(sec)	1.26	16.50	12.25	8.95	3.53	4.03	2.08	7.39	13.67	2.30
F8										
Mean	**2.22E-16**	2.19E + 01	1.91E + 02	5.94E + 01	6.65E + 01	**2.22E-16**	**2.22E-16**	**2.22E-16**	3.74E + 01	**2.22E-16**
Std	2.47E-32	7.14E + 00	1.00E + 01	4.04E + 00	3.02E + 01	2.47E-32	2.47E-32	2.47E-32	8.80E + 00	2.47E-32
Time(sec)	1.24	16.56	12.58	9.00	3.52	4.03	2.06	7.37	13.69	2.29
F9										
Mean	6.13E + 00	3.84E + 01	2.48E + 01	1.39E + 01	2.97E + 00	2.58E + 01	2.36E + 01	**4.62E-06**	2.87E + 01	2.83E + 01
Std	7.83E + 00	6.39E + 01	1.30E-01	2.84E + 00	6.25E + 00	7.28E-01	2.45E-01	7.74E-06	1.35E + 01	3.60E-01
Time(sec)	1.18	16.28	11.95	8.50	3.39	3.97	1.99	7.11	13.46	2.18
F10										
Mean	6.06E-03	1.07E-02	1.68E-05	1.19E-06	4.56E + 01	**2.22E-16**	5.07E-04	**2.22E-16**	1.76E-02	**2.22E-16**
Std	7.67E-03	1.27E-02	1.28E-04	1.23E-06	1.39E + 01	2.47E-32	2.35E-03	2.47E-32	1.98E-02	2.47E-32
Time(sec)	1.34	16.58	12.28	8.93	3.81	4.09	2.14	7.48	13.82	2.40
F11										
Mean	**3.82E-04**	2.29E + 03	6.82E + 03	2.86E + 03	6.84E + 03	5.95E + 03	2.32E + 02	**5.75E-04**	5.66E + 03	6.51E + 03
Std	4.04E-12	4.20E + 02	3.07E + 02	1.59E + 02	5.42E + 02	6.61E + 02	5.59E + 02	3.39E-04	6.23E + 02	8.21E + 02
Time(sec)	2.56	17.25	13.49	10.98	4.38	4.93	2.83	9.19	14.47	3.10
F12										
Mean	**2.21E-14**	**2.22E-16**	5.71E + 00	**2.22E-16**	**2.22E-16**	1.21E + 01	**2.22E-16**	**2.22E-16**	1.30E + 01	1.09E + 00
Std	1.47E-14	2.47E-32	1.13E + 00	2.47E-32	2.47E-32	5.61E + 00	2.47E-32	2.47E-32	3.76E + 00	2.40E + 00
Time(sec)	27.22	41.98	38.77	60.86	29.93	30.01	27.62	74.14	39.85	27.94

Table 8 Results for the multimodal benchmark functions for D = 50

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F7										
Mean	**8.73E-14**	1.59E-10	5.82E-01	1.75E + 00	1.34E + 01	**9.53E-15**	**2.75E-15**	**2.22E-16**	2.50E + 00	**2.55E-15**
Std	1.50E-14	7.54E-10	3.11E-01	2.24E-01	1.06E + 00	2.94E-15	2.21E-15	2.47E-32	8.96E-01	1.53E-15
Time(sec)	1.30	17.06	12.51	11.06	3.89	5.61	2.33	8.84	17.74	2.74
F8										
Mean	**2.48E-16**	7.05E + 01	4.36E + 02	1.37E + 02	1.04E + 02	**2.22E-16**	**2.22E-16**	**2.22E-16**	9.18E + 01	**2.22E-16**
Std	1.99E-16	1.87E + 01	1.56E + 01	1.26E + 01	3.29E + 01	2.47E-32	2.47E-32	2.47E-32	1.79E + 01	2.47E-32
Time(sec)	1.28	17.05	13.00	11.22	3.89	5.62	2.31	8.81	17.83	2.73
F9										
Mean	5.77E + 01	1.25E + 02	1.19E + 03	3.97E + 01	2.71E + 01	4.58E + 01	4.32E + 01	**6.59E-06**	4.76E + 01	4.85E + 01
Std	2.86E + 01	3.33E + 02	3.03E + 02	2.25E + 00	7.00E + 00	7.80E-01	5.63E + 00	9.95E-06	5.72E + 00	3.68E-01
Time(sec)	1.19	16.66	12.20	10.38	3.69	5.55	2.22	8.46	17.46	2.58
F10										
Mean	9.73E-03	5.05E-03	3.50E-01	9.05E-05	9.28E + 01	1.25E-04	7.11E-04	**2.22E-16**	1.48E-02	**2.22E-16**
Std	9.00E-03	5.52E-03	9.51E-02	4.41E-05	2.60E + 01	9.62E-04	3.57E-03	2.47E-32	1.90E-02	2.47E-32
Time(sec)	1.44	17.20	12.70	11.15	4.37	5.72	2.44	9.03	18.06	2.92
F11										
Mean	**6.36E-04**	5.70E + 03	1.35E + 04	6.34E + 03	1.32E + 04	1.14E + 04	5.96E + 02	1.03E-03	1.04E + 04	1.17E + 04
Std	9.75E-12	6.17E + 02	4.12E + 02	2.64E + 02	8.87E + 02	1.02E + 03	1.42E + 03	9.19E-04	8.75E + 02	1.41E + 03
Time(sec)	3.14	18.27	14.27	14.17	5.27	7.09	3.59	11.85	19.18	4.05
F12										
Mean	2.25E-05	**2.22E-16**	2.26E + 01	**2.22E-16**	**2.22E-16**	3.09E + 01	**2.22E-16**	**2.22E-16**	3.63E + 01	2.85E + 00
Std	1.35E-04	2.47E-32	1.65E + 00	2.47E-32	2.47E-32	1.35E + 01	2.47E-32	2.47E-32	5.53E + 00	5.84E + 00
Time(sec)	44.38	59.33	56.30	97.10	47.88	48.84	44.81	119.71	61.46	45.40

Table 9 Results for the multimodal benchmark functions for D = 100

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F7										
Mean	3.06E-04	2.72E-01	1.66E + 01	2.74E + 00	1.37E + 01	**1.81E-14**	**2.86E-15**	**2.22E-16**	1.08E + 01	**2.55E-15**
Std	1.56E-04	1.33E + 00	2.46E-01	1.44E-01	7.14E-01	3.53E-15	2.34E-15	2.47E-32	7.74E-01	1.53E-15
Time(sec)	1.45	18.13	13.73	16.36	4.69	9.60	3.10	12.48	31.31	3.86
F8										
Mean	3.13E-06	2.51E + 02	1.14E + 03	3.86E + 02	1.96E + 02	**2.22E-16**	**2.22E-16**	**2.22E-16**	2.84E + 02	**2.22E-16**
Std	3.50E-06	5.01E + 01	3.90E + 01	2.73E + 01	5.78E + 01	2.47E-32	2.47E-32	2.47E-32	3.36E + 01	2.47E-32
Time(sec)	1.41	18.14	13.75	16.70	4.68	9.64	3.09	12.49	31.30	3.86
F9										
Mean	1.78E + 02	3.32E + 02	1.60E + 05	9.56E + 01	8.83E + 01	9.62E + 01	9.12E + 01	**1.84E-05**	1.23E + 03	9.89E + 01
Std	4.37E + 01	6.18E + 02	2.02E + 04	5.76E-01	8.92E + 00	7.80E-01	1.69E + 01	2.88E-05	2.93E + 02	2.45E-01
Time(sec)	1.26	17.57	13.07	15.14	4.29	9.53	2.93	11.87	30.68	3.57
F10										
Mean	1.88E-02	3.03E + 00	3.29E + 02	4.97E-02	2.28E + 02	**2.22E-16**	**2.22E-16**	**2.22E-16**	3.36E + 01	**2.22E-16**
Std	1.76E-02	1.63E + 01	2.49E + 01	1.97E-02	4.87E + 01	2.47E-32	2.47E-32	2.47E-32	7.36E + 00	2.47E-32
Time(sec)	1.73	18.65	14.44	16.93	5.67	9.84	3.33	12.96	32.14	4.33
F11										
Mean	**1.30E-03**	1.68E + 04	3.09E + 04	1.67E + 04	3.06E + 04	2.48E + 04	8.34E + 02	**2.06E-03**	2.36E + 04	2.34E + 04
Std	2.44E-05	1.37E + 03	1.20E + 03	4.01E + 02	2.00E + 03	1.68E + 03	1.67E + 03	1.50E-03	1.52E + 03	2.14E + 03
Time(sec)	4.57	20.80	16.36	22.16	7.31	12.51	5.61	18.49	34.10	6.46
F12										
Mean	1.45E-02	**2.22E-16**	7.57E + 01	**2.22E-16**	**2.22E-16**	7.15E + 01	**2.22E-16**	**2.22E-16**	1.11E + 02	6.84E + 00
Std	9.80E-03	2.47E-32	2.79E + 00	2.47E-32	2.47E-32	4.63E + 01	2.47E-32	2.47E-32	6.87E + 00	1.53E + 01
Time(sec)	87.42	102.70	100.16	187.52	91.23	95.77	87.90	233.46	132.91	89.20

Table 10 Results for the composite benchmark functions for D = 30

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F13										
Mean	**2.38E + 03**	**2.37E + 03**	2.49E + 03	2.40E + 03	2.54E + 03	**2.37E + 03**	2.57E + 03	2.51E + 03	2.47E + 03	2.63E + 03
Std	4.57E + 01	1.61E + 01	1.24E + 01	6.04E + 01	4.46E + 01	1.93E + 01	5.97E + 01	3.98E + 01	3.29E + 01	2.39E + 01
Time(sec)	4.77	19.93	16.23	15.85	7.00	7.39	5.39	15.11	17.08	5.64
F14										
Mean	3.67E + 03	3.71E + 03	3.33E + 03	**2.51E + 03**	6.99E + 03	4.10E + 03	6.42E + 03	5.53E + 03	3.82E + 03	7.61E + 03
Std	1.39E + 03	1.46E + 03	5.95E + 02	7.37E + 02	1.48E + 03	1.62E + 03	2.17E + 03	2.15E + 03	2.08E + 03	1.40E + 03
Time(sec)	5.88	20.85	17.56	18.23	8.13	8.47	6.45	17.86	18.07	6.74
F15										
Mean	**2.72E + 03**	2.79E + 03	2.85E + 03	2.77E + 03	3.18E + 03	2.73E + 03	3.05E + 03	3.02E + 03	3.04E + 03	3.13E + 03
Std	2.38E + 01	3.49E + 01	9.48E + 00	2.42E + 01	1.20E + 02	3.05E + 01	1.05E + 02	8.94E + 01	7.05E + 01	6.66E + 01
Time(sec)	6.55	21.51	18.23	19.55	8.80	9.13	7.12	19.10	18.78	7.40
F16										
Mean	3.01E + 03	2.98E + 03	3.01E + 03	**2.88E + 03**	3.38E + 03	2.90E + 03	3.18E + 03	3.28E + 03	3.18E + 03	3.36E + 03
Std	1.63E + 02	5.10E + 01	1.02E + 01	1.16E + 02	1.65E + 02	4.46E + 01	9.72E + 01	1.44E + 02	7.71E + 01	6.60E + 01
Time(sec)	6.35	21.28	17.86	18.88	8.52	8.85	6.85	18.19	18.54	7.11
F17										
Mean	**2.89E + 03**	2.91E + 03	**2.89E + 03**	2.88E + 03	3.26E + 03	2.94E + 03	2.94E + 03	2.91E + 03	2.93E + 03	4.18E + 03
Std	1.76E + 00	3.89E + 01	7.21E-02	4.26E-02	1.58E + 02	2.57E + 01	2.57E + 01	1.82E + 01	2.19E + 01	3.86E + 02
Time(sec)	6.12	21.23	17.45	18.25	8.44	8.84	6.84	18.45	18.56	7.09
F18										
Mean	**3.02E + 03**	4.65E + 03	5.59E + 03	3.13E + 03	8.26E + 03	4.39E + 03	7.63E + 03	6.65E + 03	6.23E + 03	7.94E + 03
Std	5.25E + 02	4.97E + 02	9.15E + 01	1.73E + 02	1.06E + 03	3.27E + 02	9.05E + 02	1.46E + 03	1.74E + 03	4.74E + 02
Time(sec)	7.91	23.01	19.71	22.46	10.19	10.59	8.56	22.65	20.19	8.84
F19										
Mean	3.23E + 03	3.25E + 03	**3.21E + 03**	3.22E + 03	3.52E + 03	3.23E + 03	3.34E + 03	3.30E + 03	3.53E + 03	3.78E + 03
Std	8.88E + 00	3.16E + 01	3.56E + 00	5.22E + 00	1.64E + 02	1.35E + 01	5.52E + 01	4.05E + 01	9.70E + 01	2.18E + 02
Time(sec)	9.14	23.81	20.78	24.49	11.24	11.58	9.53	24.46	21.18	9.84
F20										
Mean	**3.20E + 03**	3.31E + 03	3.22E + 03	3.20E + 03	4.16E + 03	3.33E + 03	3.28E + 03	3.24E + 03	3.26E + 03	5.22E + 03
Std	7.83E + 00	7.98E + 01	9.67E + 00	1.01E + 01	4.26E + 02	4.08E + 01	2.49E + 01	2.90E + 01	2.55E + 01	4.07E + 02
Time(sec)	7.63	22.81	19.12	21.54	9.96	10.31	8.32	22.14	20.10	8.56

Table 10 (continued)

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F21										
Mean	**3.56E + 03**	3.57E + 03	4.13E + 03	3.74E + 03	5.61E + 03	3.64E + 03	4.78E + 03	4.25E + 03	4.19E + 03	4.71E + 03
Std	1.18E + 02	1.17E + 02	1.16E + 02	6.97E + 01	6.49E + 02	1.35E + 02	3.41E + 02	3.08E + 02	2.60E + 02	2.95E + 02
Time(sec)	7.33	22.40	19.18	21.01	9.57	9.88	7.89	21.30	19.60	8.17
F22										
Mean	**8.41E + 03**	2.41E + 05	4.18E + 04	9.99E + 03	2.07E + 07	4.59E + 06	8.04E + 06	7.13E + 05	6.07E + 04	2.91E + 08
Std	1.18E + 03	7.74E + 05	2.69E + 04	1.30E + 03	2.88E + 07	3.46E + 06	6.65E + 06	3.91E + 05	3.08E + 04	2.56E + 08
Time(sec)	10.28	25.25	22.02	26.59	12.41	12.70	10.73	28.21	22.50	10.96

all functions for all dimensions and the convergence speed of ALSO is very high which is helpful for local optima avoidance.

Two samples Wilcoxon rank-sum test is conducted to statistically compare the best fitness values obtained from each benchmark function of all algorithms. The p-values calculated by the non-parametric Wilcoxon rank sum tests for the pair-wise comparison over two independent samples are performed at 5% significance level and are reported in Appendix Tables 16, 17, 18, "0" means that the performances of two algorithms are equal. "1 + " shows that the ALSO is better than the compared algorithms and "1-" indicates that the ALSO is worse than the compared algorithms. The performance of the ALSO is the number of "0" and "1 + ".

Inspecting the results in Appendix Table 16, it reveals that the ALSO gives very outstanding performance compared with other algorithms, and the performance of the ALSO is calculated as 86% for the dimension 30. Thus, it can be indicated that the superiority of the proposed algorithm is statistically significant. As per results presented in Appendix Table 17, the *p*-values show that the superiority of the ALSO is statistically significant. The performance of the proposed optimizer is calculated as 84% for the dimension 50. The *p*-values reported in Appendix Table 18 indicates that the superiority of the ALSO is statistically significant. Moreover, the performance of the proposed algorithm is calculated as 83% for the dimension 100.

4 ALSO for classical engineering design problems

In this section, well-known three engineering design problems such as tension/compression spring, welded beam, and pressure vessel are discussed. These problems have many equality and inequality constraints. Therefore, the ALSO must be adapted to find the optimal solutions for the constrained problems. Constraints in the problem restrict the search space. Search agents who are out of the feasible region follow two ways. The first is that the agents are replaced to a new random position. However, this process causes unnecessary cycles when the search agents locate in the infeasible region. The second is that penalizing the infeasible solutions by reducing their fitness value. There are many different approaches to penalize the infeasible solutions (Yeniay 2005). In this study, the penalty function approaches are used to handle the constraints

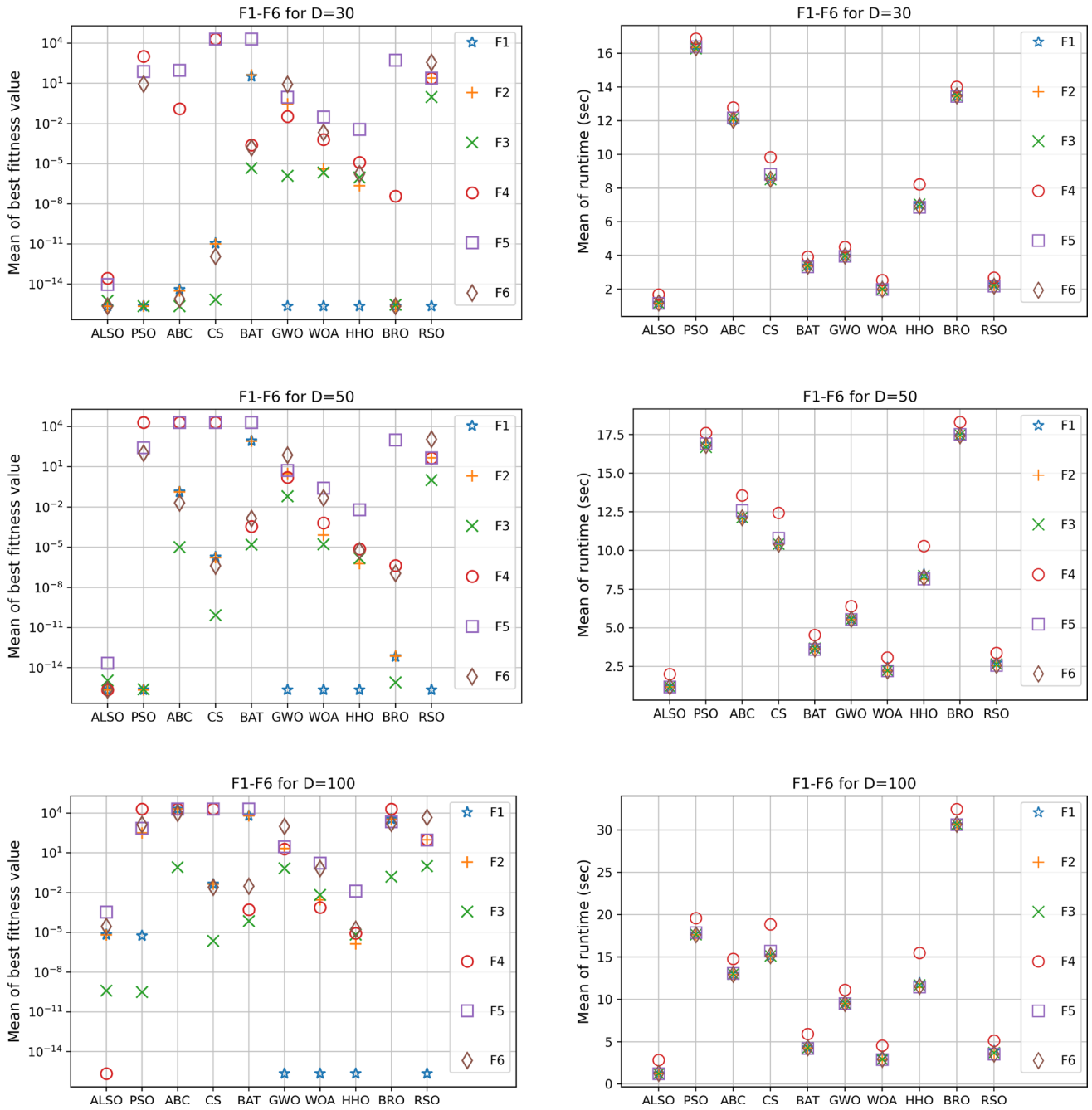

Fig. 8 Simulation results of the unimodal benchmark functions according to best fitness value and runtime of the algorithms

and a constant penalty value is defined. If the search agents locate in the infeasible region, it is added to objective value. But this situation causes discontinuities of the function. Therefore, the feasible solution can be in a ridge region, and it causes the difficulty of the single-dimensional searches. Because of this, the resultant of the multiple dimensional searches is used. The ALSO is run 30 times for the engineering problems and the best results are chosen for comparison.

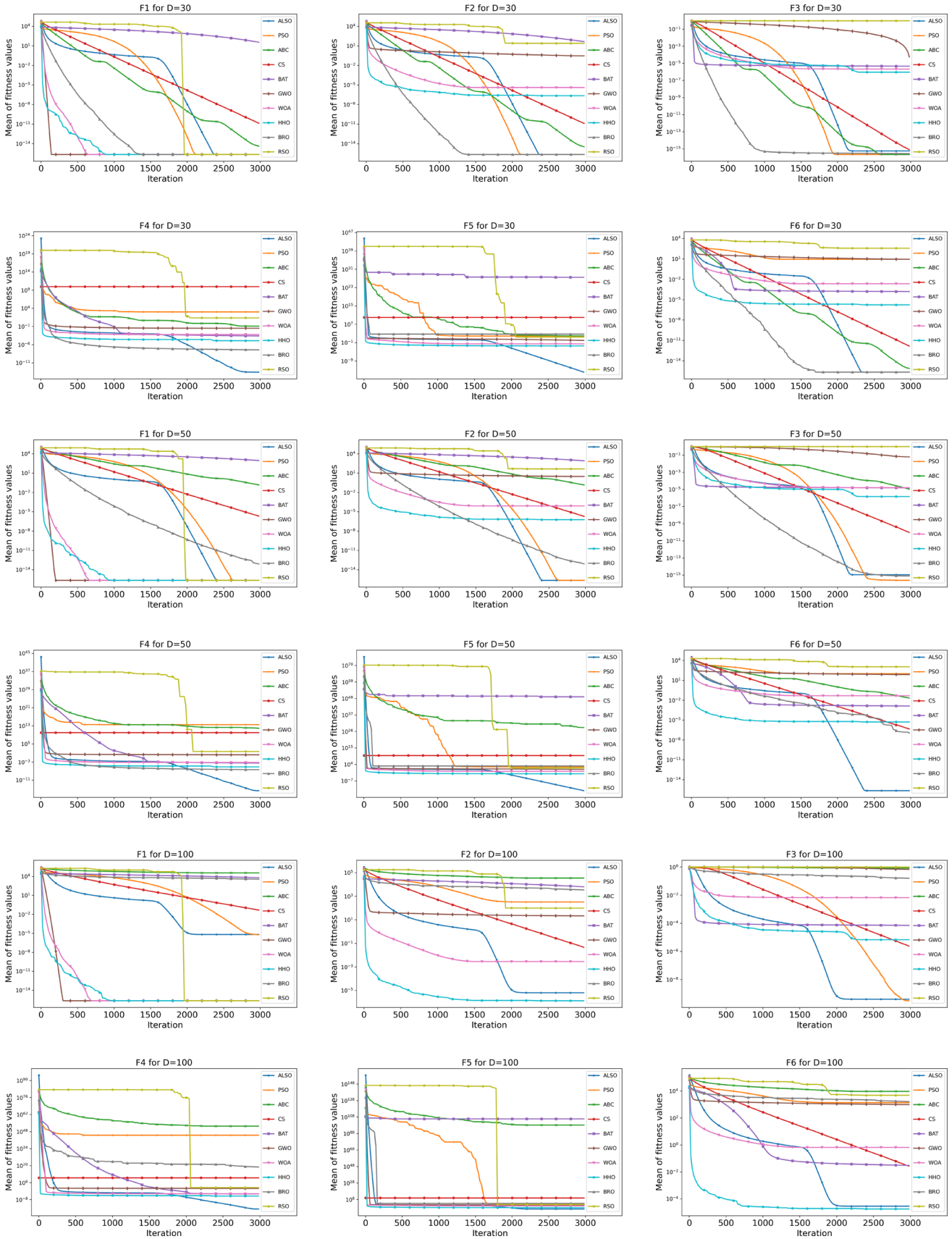

Fig. 9 Convergence of algorithms on the unimodal test functions for $D = 30, 50, 100$

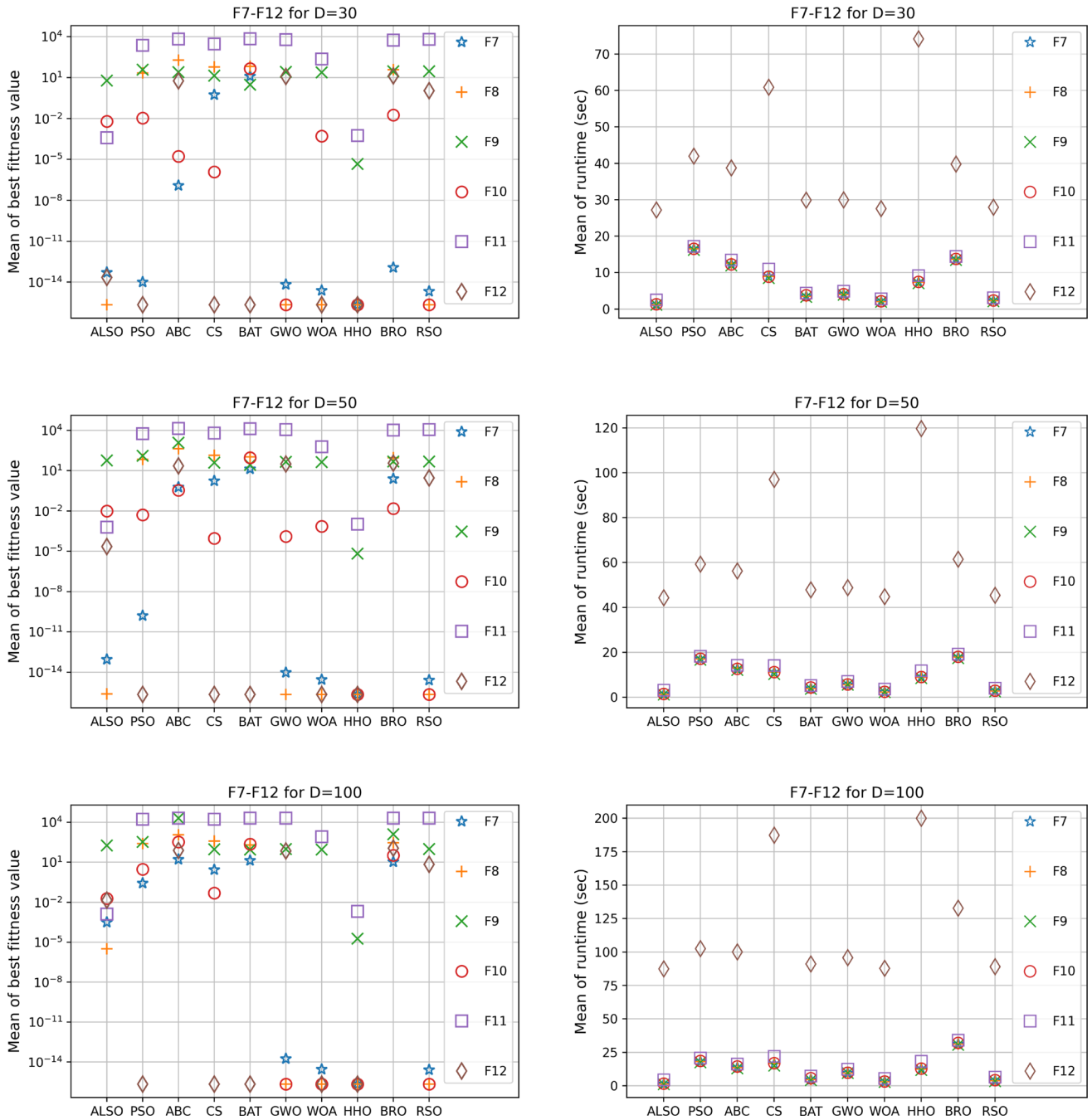

Fig. 10 Simulation results of the multimodal benchmark functions according to best fitness value and runtime of the algorithms

4.1 Tension/compression spring design problem

The tension/compression spring design problem is widely used in engineering. The objective of this problem is to minimize the weight of a tension/compression spring as illustrated in Fig. 16.

The problem consists of three design variables: wire diameter (d), mean coil diameter (D) and number of active

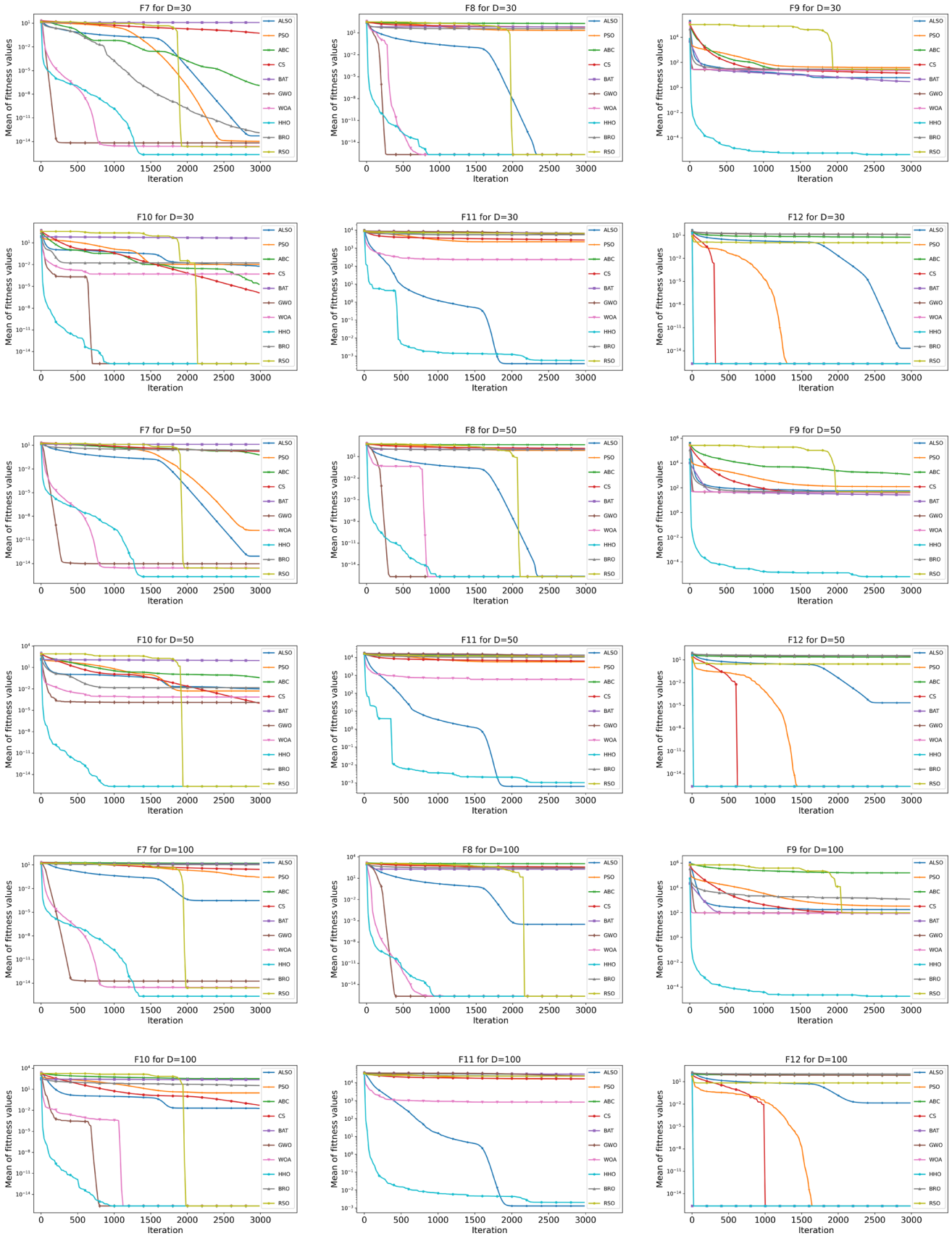

Fig. 11 Convergence of algorithms on the multimodal test functions for $D = 30, 50, 100$

Table 11 Results for the composite benchmark functions for D = 50

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F13										
Mean	2.51E + 03	**2.48E + 03**	2.74E + 03	2.61E + 03	2.79E + 03	**2.48E + 03**	2.85E + 03	2.78E + 03	2.66E + 03	3.03E + 03
Std	5.79E + 01	4.50E + 01	1.35E + 01	3.01E + 01	8.23E + 01	3.87E + 01	8.22E + 01	7.61E + 01	5.91E + 01	4.62E + 01
Time(sec)	8.03	23.80	19.99	24.55	10.77	12.30	8.94	24.45	24.48	9.39
F14										
Mean	**7.54E + 03**	8.46E + 03	1.58E + 04	9.25E + 03	1.16E + 04	8.63E + 03	1.16E + 04	1.06E + 04	1.00E + 04	1.56E + 04
Std	7.83E + 02	1.28E + 03	5.91E + 02	1.26E + 03	1.48E + 03	1.95E + 03	1.24E + 03	1.12E + 03	1.12E + 03	8.68E + 02
Time(sec)	9.79	25.68	21.99	28.49	12.68	14.10	10.75	29.21	26.38	11.25
F15										
Mean	2.96E + 03	3.13E + 03	3.16E + 03	3.06E + 03	3.93E + 03	**2.91E + 03**	3.61E + 03	3.57E + 03	3.61E + 03	3.74E + 03
Std	4.18E + 01	9.28E + 01	1.46E + 01	3.34E + 01	2.33E + 02	5.26E + 01	1.61E + 02	1.23E + 02	1.80E + 02	1.03E + 02
Time(sec)	11.27	26.89	23.55	31.44	14.23	15.64	12.26	32.24	27.81	12.72
F16										
Mean	3.53E + 03	3.33E + 03	3.30E + 03	3.24E + 03	4.16E + 03	**3.09E + 03**	3.67E + 03	4.02E + 03	3.66E + 03	4.12E + 03
Std	1.88E + 02	1.08E + 02	1.33E + 01	3.13E + 01	2.24E + 02	8.96E + 01	1.55E + 02	1.63E + 02	1.43E + 02	1.65E + 02
Time(sec)	11.04	26.48	22.94	30.38	13.71	15.15	11.78	30.48	27.40	12.26
F17										
Mean	**3.02E + 03**	3.16E + 03	3.10E + 03	2.99E + 03	6.76E + 03	3.32E + 03	3.15E + 03	3.13E + 03	3.14E + 03	1.15E + 04
Std	2.61E + 01	1.22E + 02	1.90E + 01	1.58E + 01	1.13E + 03	1.58E + 02	3.92E + 01	3.38E + 01	2.14E + 01	9.97E + 02
Time(sec)	11.24	26.94	23.25	30.75	14.10	15.55	12.20	31.31	27.88	12.63
F18										
Mean	**5.01E + 03**	6.43E + 03	8.02E + 03	5.48E + 03	1.37E + 04	5.79E + 03	1.29E + 04	9.99E + 03	1.11E + 04	1.32E + 04
Std	1.51E + 03	8.78E + 02	1.25E + 02	1.19E + 03	1.51E + 03	4.34E + 02	1.49E + 03	2.42E + 03	1.13E + 03	8.33E + 02
Time(sec)	14.08	29.85	26.46	37.16	17.05	18.48	15.05	38.66	30.61	15.52
F19										
Mean	**3.43E + 03**	3.58E + 03	**3.42E + 03**	3.46E + 03	4.40E + 03	3.47E + 03	4.20E + 03	3.89E + 03	4.89E + 03	5.08E + 03
Std	5.64E + 01	1.38E + 02	3.31E + 01	3.93E + 01	3.04E + 02	6.68E + 01	3.63E + 02	1.83E + 02	3.15E + 02	4.78E + 02
Time(sec)	16.27	31.46	28.44	41.34	19.07	20.46	17.06	43.68	32.60	17.60
F20										
Mean	**3.29E + 03**	3.89E + 03	3.37E + 03	3.26E + 03	7.44E + 03	3.85E + 03	3.44E + 03	3.36E + 03	3.58E + 03	7.80E + 03
Std	1.86E + 01	6.55E + 02	1.51E + 01	1.54E + 00	1.14E + 03	2.78E + 02	6.97E + 01	3.42E + 01	3.91E + 01	6.30E + 02
Time(sec)	14.22	29.86	26.29	36.62	17.06	18.41	15.09	38.50	30.86	15.49

Table 11 (continued)

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F21										
Mean	**3.97E + 03**	4.11E + 03	5.23E + 03	4.51E + 03	1.24E + 04	4.22E + 03	7.01E + 03	5.24E + 03	5.69E + 03	1.35E + 04
Std	2.13E + 02	3.42E + 02	1.62E + 02	1.28E + 02	4.14E + 03	2.37E + 02	8.05E + 02	5.21E + 02	4.06E + 02	2.46E + 03
Time(sec)	12.07	27.63	24.31	32.65	14.90	16.25	12.90	34.10	28.49	13.36
F22										
Mean	**8.16E + 05**	4.83E + 06	6.79E + 06	4.05E + 06	2.58E + 08	7.09E + 07	8.25E + 07	1.28E + 07	2.19E + 07	2.33E + 09
Std	7.64E + 04	4.78E + 06	4.71E + 06	9.74E + 05	1.37E + 08	2.46E + 07	2.83E + 07	3.34E + 06	2.21E + 06	1.04E + 09
Time(sec)	17.13	32.40	29.04	41.87	19.71	20.96	17.63	45.60	33.39	18.05

coils (N). The mathematical formulation of this problem is as follows (Mirjalili and Lewis 2016):

$$\text{Consider}: \ \mathbf{x} = [x_1 \quad x_2 \quad x_3] = [d \quad D \quad N]$$

$$\text{Minimize}: \ f(\mathbf{x}) = (x_3 + 2)x_2x_1^2$$

$$\text{Subject to}: \ g_1(\mathbf{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0,$$

$$g_2(\mathbf{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \leq 0,$$

$$g_3(\mathbf{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0,$$

$$g_4(\mathbf{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0,$$

$$\text{Variable bound}: \ x_1 \in [0.05, 2.00]$$
$$x_2 \in [0.25, 1.30]$$
$$x_3 \in [2.00, 15.0]$$

The ALSO algorithm is applied to the tension/compression design problem with 50 search agents and maximum iteration is set to 1000. The results obtained by the ALSO is compared with the GA (Coello 2000), PSO (He and Wang 2007), BAT (Gandomi et al. 2013), GWO (Mirjalili et al. 2014), WOA (Mirjalili and Lewis 2016), CSA (Askarzadeh 2016), and LS-II (Camarena et al. 2018). Note that the results handled in original papers have been considered for other methods. Table 13 outlines the comparison results of the algorithms.

According to Table 13, the ALSO, BAT, CSA, and LS-II give the best result for tension/compression spring design problem. However, the BAT does not satisfy the first constraint, CSA does not provide the second constraint, and LS-II does not satisfy first and second constraints. The ALSO is outstanding algorithm and satisfies all constraints.

4.2 Welded beam design problem

The main goal of welded beam design problem illustrated in Fig. 17 is to find the minimum fabrication cost. This problem has two linear and five nonlinear inequality constraints which are shear stress (τ), bending stress in the beam (θ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints (Mirjalili et al. 2014; Mirjalili and Lewis 2016; Coello 2000; He and Wang 2007; Gandomi et al. 2013; Askarzadeh 2016).

Welded beam design problem has four decision variables, namely thickness of weld (h), length of attached part of bar (l), the height of the bar (t), and thickness of the bar (b). The problem is formulated as follows (Mirjalili and Lewis 2016):

Table 12 Results for the composite benchmark functions for D = 100

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F13										
Mean	3.01E + 03	2.96E + 03	3.47E + 03	3.21E + 03	4.04E + 03	**2.86E + 03**	3.90E + 03	3.82E + 03	3.67E + 03	4.55E + 03
Std	6.53E + 01	7.90E + 01	3.25E + 01	6.72E + 01	2.35E + 02	1.14E + 02	1.69E + 02	1.55E + 02	1.26E + 02	1.22E + 02
Time(sec)	22.03	39.01	34.54	57.69	25.59	30.26	23.60	61.25	55.50	24.43
F14										
Mean	**1.54E + 04**	1.71E + 04	3.36E + 04	2.24E + 04	2.31E + 04	1.79E + 04	2.52E + 04	2.30E + 04	2.17E + 04	3.26E + 04
Std	9.06E + 02	1.34E + 03	5.56E + 02	4.46E + 02	3.33E + 03	2.80E + 03	2.17E + 03	1.58E + 03	1.50E + 03	1.56E + 03
Time(sec)	25.32	42.65	38.45	65.33	29.17	33.79	27.27	70.82	59.64	28.02
F15										
Mean	**3.19E + 03**	4.19E + 03	3.90E + 03	3.72E + 03	5.58E + 03	3.40E + 03	4.67E + 03	4.64E + 03	5.14E + 03	5.65E + 03
Std	4.89E + 01	2.15E + 02	2.48E + 01	5.44E + 01	3.36E + 02	8.79E + 01	2.14E + 02	2.10E + 02	2.73E + 02	3.11E + 02
Time(sec)	30.44	47.15	43.71	75.88	34.26	39.14	32.56	81.23	65.06	33.36
F16										
Mean	**3.90E + 03**	5.09E + 03	4.35E + 03	4.21E + 03	7.93E + 03	3.93E + 03	5.95E + 03	5.92E + 03	5.30E + 03	8.09E + 03
Std	6.91E + 01	3.01E + 02	2.63E + 01	5.96E + 01	8.45E + 02	9.96E + 01	4.40E + 02	3.73E + 02	2.54E + 02	4.82E + 02
Time(sec)	29.85	46.19	42.59	73.57	33.57	38.22	31.58	78.91	64.00	32.39
F17										
Mean	**3.29E + 03**	4.29E + 03	8.96E + 03	3.32E + 03	1.51E + 04	5.07E + 03	3.85E + 03	3.60E + 03	5.79E + 03	1.83E + 04
Std	5.06E + 01	7.37E + 02	5.51E + 02	3.20E + 01	2.74E + 03	3.43E + 02	1.12E + 02	6.41E + 01	4.00E + 02	1.25E + 03
Time(sec)	32.81	49.80	45.70	79.05	36.88	41.26	34.65	86.22	67.20	35.32
F18										
Mean	**1.27E + 04**	1.77E + 04	1.72E + 04	1.49E + 04	3.96E + 04	**1.25E + 04**	3.23E + 04	2.49E + 04	3.04E + 04	3.71E + 04
Std	2.42E + 03	2.84E + 03	2.45E + 02	6.29E + 02	4.85E + 03	8.82E + 02	3.56E + 03	1.64E + 03	1.92E + 03	3.66E + 03
Time(sec)	38.47	55.36	51.82	91.64	42.84	47.16	40.51	101.23	73.46	41.28
F19										
Mean	**3.60E + 03**	3.89E + 03	4.18E + 03	3.70E + 03	6.70E + 03	3.82E + 03	4.95E + 03	4.29E + 03	6.02E + 03	9.03E + 03
Std	6.43E + 01	2.31E + 02	7.53E + 01	5.92E + 01	9.76E + 02	9.51E + 01	5.27E + 02	3.06E + 02	7.86E + 02	7.31E + 02
Time(sec)	44.87	61.21	58.11	104.47	49.07	53.41	46.76	115.51	83.31	47.53
F20										
Mean	**3.42E + 03**	7.18E + 03	1.11E + 04	3.45E + 03	2.15E + 04	6.23E + 03	4.04E + 03	3.62E + 03	7.72E + 03	1.98E + 04
Std	2.90E + 01	2.62E + 03	6.32E + 02	4.35E + 01	2.81E + 03	8.69E + 02	1.28E + 02	4.61E + 01	6.37E + 02	2.22E + 03
Time(sec)	40.98	57.83	53.86	95.67	45.01	49.30	42.72	106.47	75.81	43.36

Table 12 (continued)

Function	ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
F21										
Mean	**6.32E + 03**	6.85E + 03	1.00E + 04	7.85E + 03	2.32E + 04	7.38E + 03	1.39E + 04	9.06E + 03	1.21E + 04	7.63E + 04
Std	4.63E + 02	6.25E + 02	2.53E + 02	1.84E + 02	5.77E + 03	4.65E + 02	1.56E + 03	5.72E + 02	9.70E + 02	3.96E + 04
Time(sec)	29.39	46.27	42.55	72.90	33.40	37.78	31.15	79.42	63.48	31.89
F22										
Mean	**1.76E + 04**	9.07E + 08	1.84E + 06	1.00E + 10	7.42E + 09	3.03E + 08	2.86E + 08	2.97E + 07	3.00E + 08	3.01E + 10
Std	6.65E + 03	9.56E + 08	6.61E + 05	0.00E + 00	3.52E + 09	2.52E + 08	1.32E + 08	8.59E + 06	8.65E + 07	2.94E + 09
Time(sec)	38.87	55.61	51.89	91.33	43.18	47.16	40.60	102.87	73.51	41.29

Consider: $\mathbf{x} = [x_1 \quad x_2 \quad x_3 \quad x_4] = [h \quad l \quad t \quad b]$

Minimize: $f(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

Subject to: $g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{\max} \leq 0,$

$g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{\max} \leq 0,$

$g_3(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{\max} \leq 0,$

$g_4(\mathbf{x}) = x_1 - x_4 \leq 0,$

$g_5(\mathbf{x}) = P - P_c(\mathbf{x}) \leq 0,$

$g_6(\mathbf{x}) = 0.125 - x_1 \leq 0$

$g_7(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) - 5.00 \leq 0,$

Variable bound: $x_1 \in [0.1, 2]$

$x_2 \in [0.1, 10]$

$x_3 \in [0.1, 10]$

$x_4 \in [0.1, 2]$

where

$$\tau(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right),$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$\sigma(\mathbf{x}) = \frac{6PL}{x_4x_3^2}, \delta(\mathbf{x}) = \frac{6PL^3}{Ex_3^2x_4}$$

$$P_c(\mathbf{x}) = \frac{4.013E\sqrt{x_3^2x_4^6}}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$P = 6000lb, L = 14in., \delta_{max} = 0.25in.,$

$E = 30000000psi, G = 12000000psi,$

$\tau_{max} = 13600psi, \sigma_{max} = 30000psi.$

The ALSO is carried out on welded beam design problem and compared with the GA (Coello 2000), PSO (He and Wang 2007), BAT (Gandomi et al. 2013), GWO (Mirjalili et al. 2014), WOA (Mirjalili and Lewis 2016), CSA (Askarzadeh 2016), and LS-II (Camarena et al. 2018).

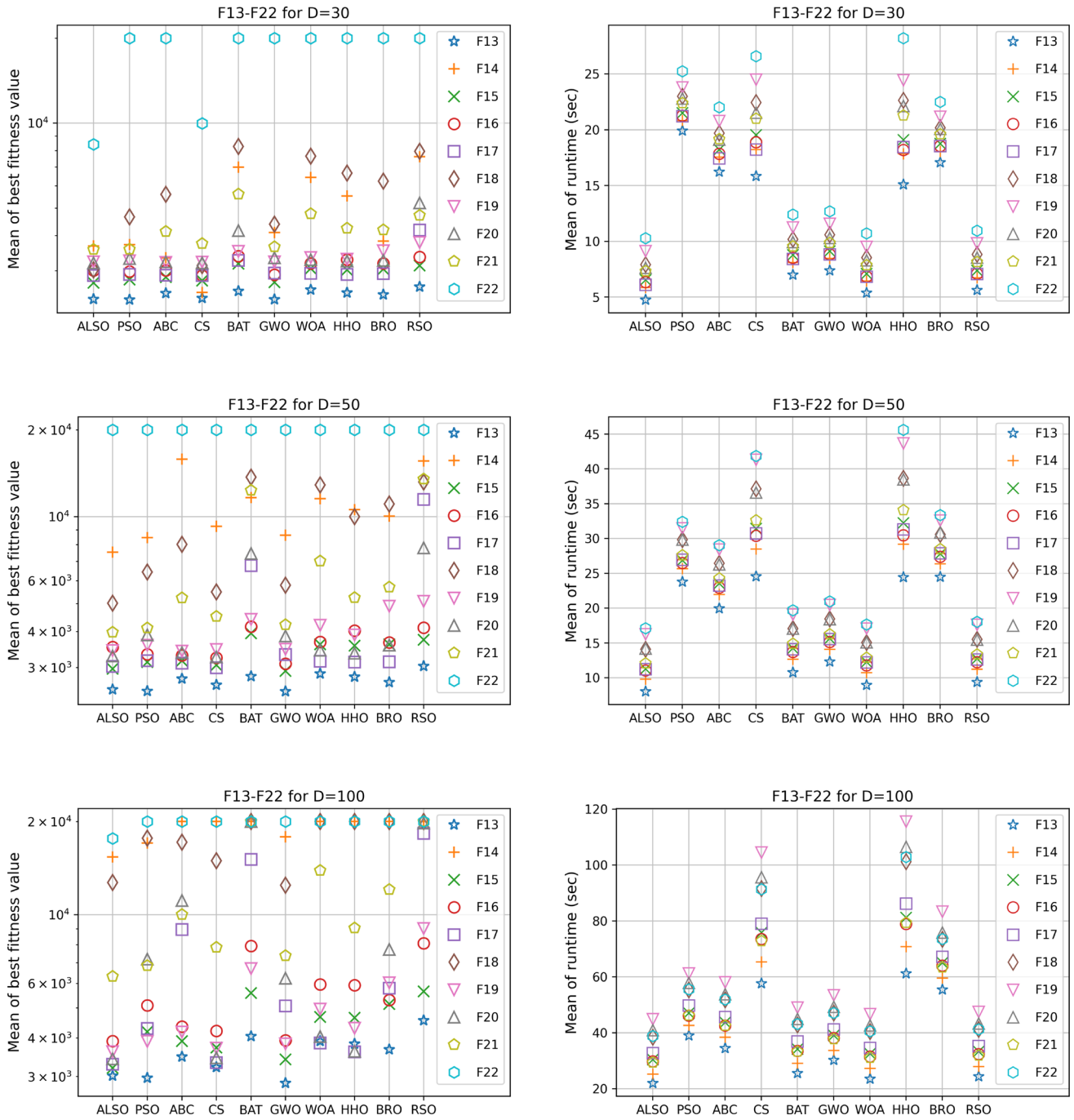

Fig. 12 Simulation results of the composite benchmark functions according to best fitness value and runtime of the algorithms

Table 14 represents the optimal results of the ALSO and reported algorithms.

Table 14 concludes that the ALSO outperforms all the other algorithms. In addition, the BAT does not provide the fifth constraint and LS-II does not satisfy the fourth constraint.

4.3 Pressure vessel design problem

Another well-known engineering optimization task is the pressure vessel design problem which goals to minimize the consisting of material, forming, and welding of a cylindrical vessel as in Fig. 18. This problem has a

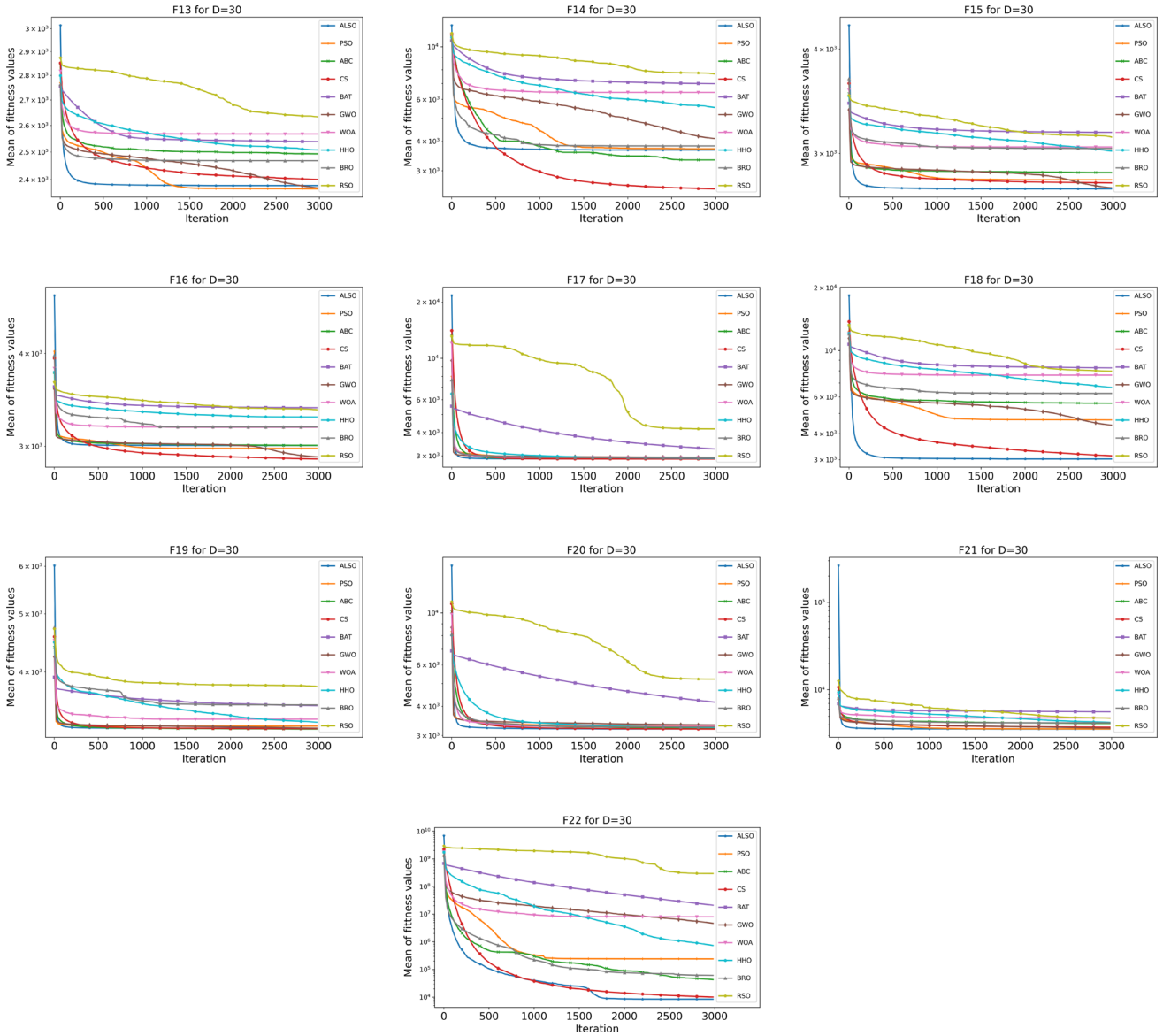

Fig. 13 Convergence of algorithms on the composite test functions for $D = 30$

nonlinear and three linear constraints. Both ends of the vessel are capped, and the head has a hemispherical shape (Mirjalili et al. 2014; Mirjalili and Lewis 2016; Coello 2000; He and Wang 2007; Gandomi et al. 2013; Askarzadeh 2016).

The decision variables are the thickness of the shell (T_s), thickness of the head (T_h), inner radius (R), length of the cylindrical section without considering the head (L). The problem can be stated as follows (Mirjalili and Lewis 2016):

$$\text{Consider: } \mathbf{x} = [x_1 \quad x_2 \quad x_3 \quad x_4] = [T_s \quad T_h \quad R \quad L]$$

$$\text{Minimize: } f(\mathbf{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$\text{Subject to: } g_1(\mathbf{x}) = -x_1 + 0.0193x_3 \leq 0,$$

$$g_2(\mathbf{x}) = -x_3 + 0.00954x_3 \leq 0,$$

$$g_3(\mathbf{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0,$$

$$g_4(\mathbf{x}) = x_4 - 240 \leq 0,$$

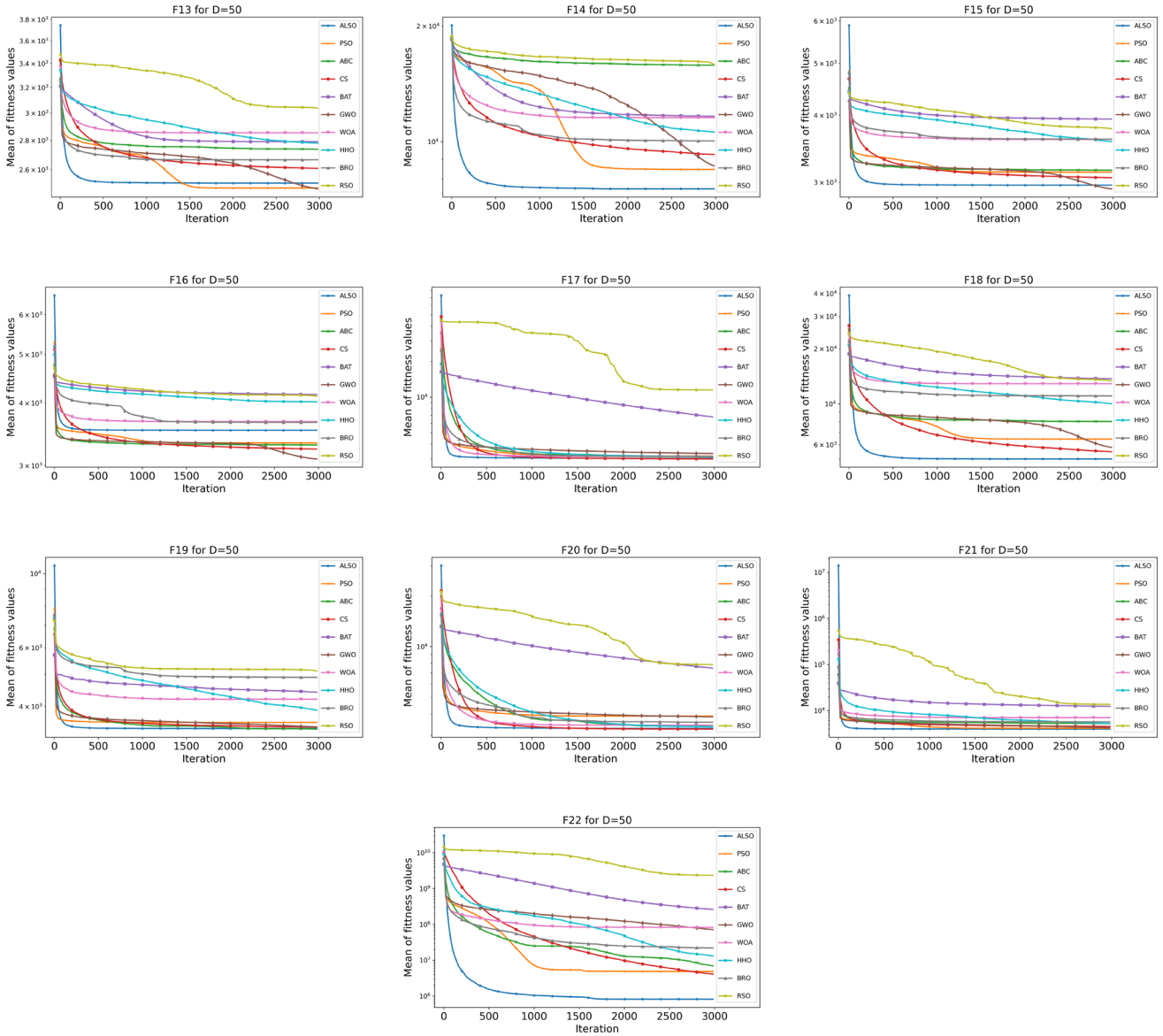

Fig. 14 Convergence of algorithms on the composite test functions for $D = 50$

Variable bound: $x_1 \in [0, 99] \times 0.0625$,
$x_2 \in [0, 99] \times 0.0625$,
$x_3 \in [10, 200]$
$x_4 \in [10, 200]$

The ALSO is performed on pressure vessel design problem and compared with the GA (Coello 2000), PSO (He and Wang 2007), BAT (Gandomi et al. 2013), GWO (Mirjalili et al. 2014), WOA (Mirjalili and Lewis 2016), CSA (Askarzadeh 2016), and LS-II (Camarena et al. 2018). Table 15 tabulates the results of the ALSO versus those other algorithms.

As can be seen in Table 15, the ALSO, BAT, GWO, CSA, and LS-II give the best result for pressure vessel design problem. In addition, the BAT does not satisfy the first constraint and the GWO does not satisfy the second boundary condition.

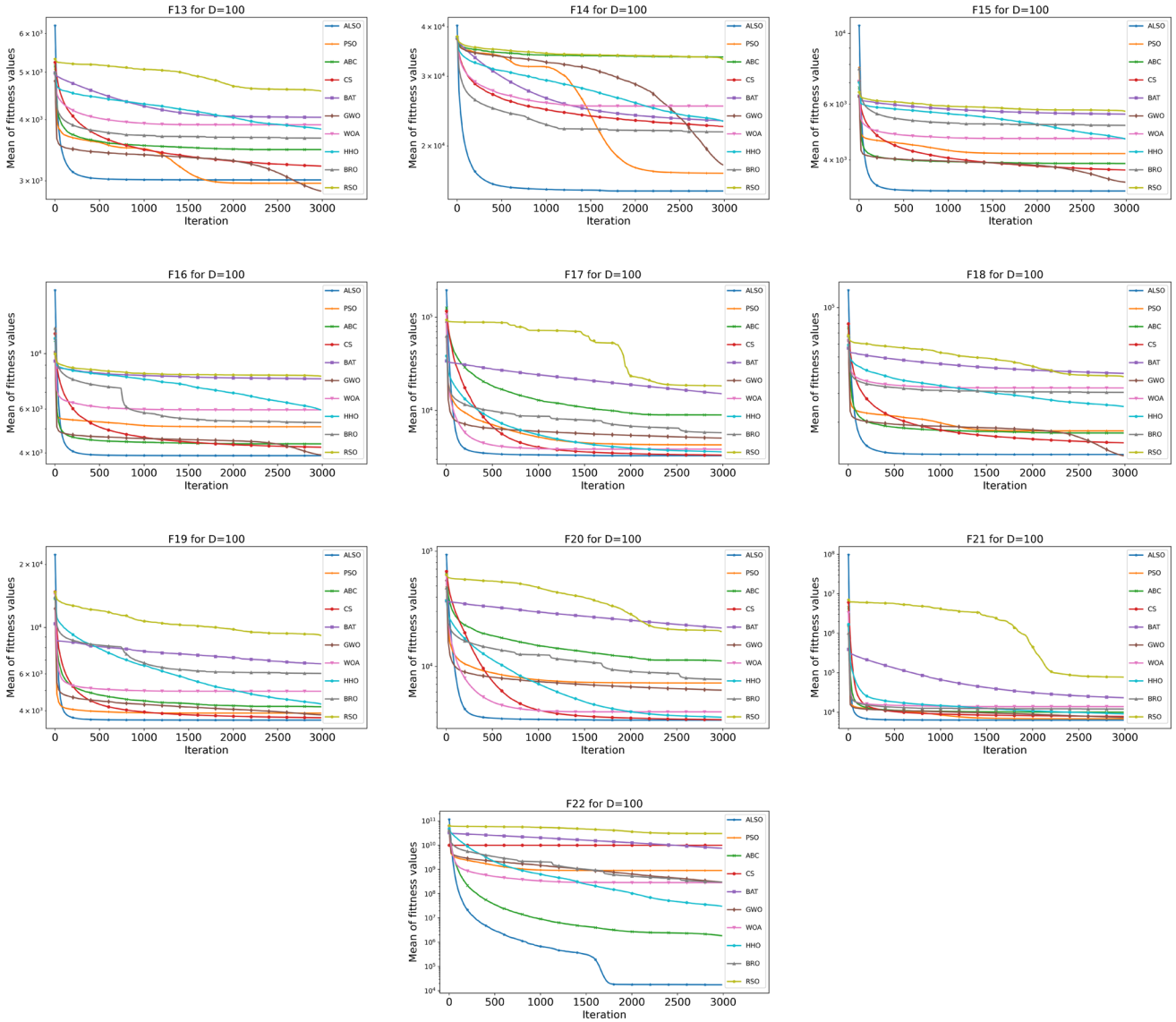

Fig. 15 Convergence of algorithms on the composite test functions for $D = 100$

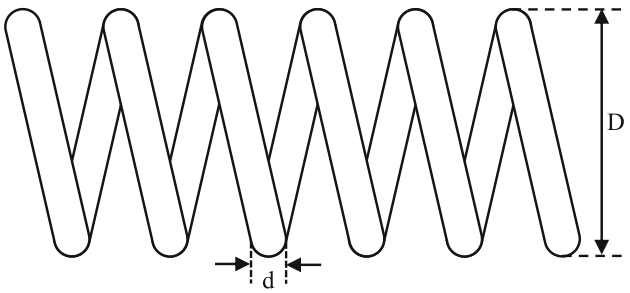

Fig. 16 The tension/compression spring

5 Conclusion

This paper proposes a swarm intelligence algorithm that models the behavior of the locust swarms to solve continuous optimization problems. The ALSO is conceptually very simple structure and easy to implement. Hence mean and standard deviation values are evidence that it presents stable solutions and achieves the global solution or reaches a near-global solution. Unlike the other methods, the proposed algorithm uses a precision function to better reveal details in the search space. The precision function is a two piecewise function that the first part provides the

Table 13 Comparison results for the tension/compression spring design problem

	ALSO	GA (Coello 2000)	PSO (He and Wang 2007)	BAT (Gandomi et al. 2013)	GWO (Mirjalili et al. 2014)	WOA (Mirjalili and Lewis 2016)	CSA (Askarzadeh 2016)	LS-II (Camarena et al. 2018)
$x_1(d)$	0.0517372	0.05148	0.051728	0.05169	0.05169	0.051207	0.051689028	0.050664
$x_2(D)$	0.357876	0.351661	0.357644	0.35673	0.356737	0.345215	0.356716954	0.329604
$x_3(N)$	11.2215	11.632201	11.244543	11.2885	11.28885	12.004032	11.2890118	4.372104
$g_1(\mathbf{x})$	− 7.48E-06	− 3.34E-03	− 8.25E-04	*1.08E-05	− 7.91E-05	− 5.64E-04	− 2.95E-09	*0.6690
$g_2(\mathbf{x})$	− 2.12E-06	− 1.10E-04	− 2.53E-05	− 2.33E-05	− 7.51E-06	− 3.70E-05	*1.94E-09	*0.9930
$g_3(\mathbf{x})$	− 4.06E + 00	− 4.03E + 00	− 4.05E + 00	− 4.05E + 00	− 4.05E + 00	− 4.03E + 00	− 4.05E + 00	− 13.9812
$g_4(\mathbf{x})$	− 7.27E-01	− 7.31E-01	− 7.27E-01	− 7.28E-01	− 7.28E-01	− 7.36E-01	− 7.28E-01	− 0.7465
$f(\mathbf{x})$	**0.012665407**	0.0127047834	0.0126747	**0.01266522**	0.012666	0.0126763	**0.0126652328**	**0.0053911**

* infeasible constraint

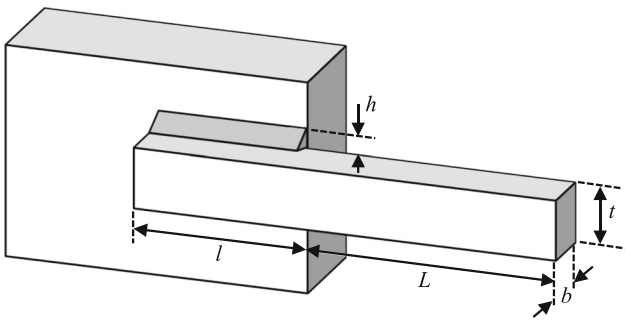

Fig. 17 The welded beam design problem

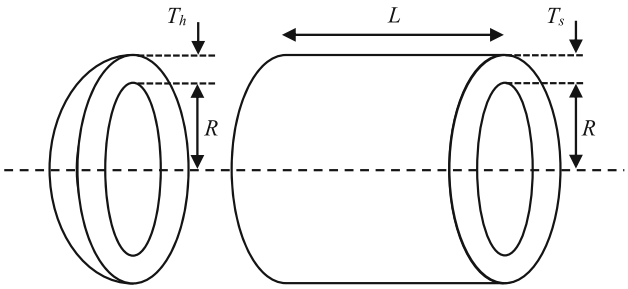

Fig. 18 The pressure vessel design problem

Table 14 Comparison results for the welded beam design problem

	ALSO	GA (Coello 2000)	PSO (He and Wang 2007)	BAT (Gandomi et al. 2013)	GWO (Mirjalili et al. 2014)	WOA (Mirjalili and Lewis 2016)	CSA (Askarzadeh 2016)	LS-II (Camarena et al. 2018)
$x_1(h)$	0.205163	0.2088	0.202369	0.2015	0.205676	0.205396	0.20572964	0.205730336
$x_2(l)$	3.25999	3.4205	3.544214	3.562	3.478377	3.484293	3.470488666	3.470474331
$x_3(t)$	9.0496	8.9975	9.04821	9.0414	9.03681	9.037426	9.03662391	9.0366238
$x_4(b)$	0.205666	0.21	0.205723	0.2057	0.205778	0.206276	0.20572964	0.205729651
$g_1(\mathbf{x})$	− 1.06E + 00	− 7.58E + 02	− 8.11E + 02	− 8.07E + 02	− 7.94E + 02	− 7.97E + 02	− 7.71E + 02	− 771.1982
$g_2(\mathbf{x})$	− 7.67E + 01	− 3.54E + 02	− 7.58E + 01	− 2.74E + 01	− 8.29E + 00	− 8.48E + 01	− 2.31E-06	− 9.0251e-04
$g_3(\mathbf{x})$	− 2.36E-01	− 2.36E-01	− 2.36E-01	− 2.36E-01	− 2.36E-01	− 2.36E-01	− 2.36E-01	− 0.0540
$g_4(\mathbf{x})$	− 5.03E-04	− 1.20E-03	− 3.35E-03	− 4.20E-03	− 1.02E-04	− 8.80E-04	0.00E + 00	*6.8500e-07
$g_5(\mathbf{x})$	− 8.86E-02	− 3.63E + 02	− 4.47E + 00	*5.09E-01	− 4.31E + 00	− 4.83E + 01	− 1.24E-06	− 9.3297e-04
$g_6(\mathbf{x})$	− 8.02E-02	− 8.38E-02	− 7.74E-02	− 7.65E-02	− 8.07E-02	− 8.04E-02	− 8.07E-02	− 0.0807
$g_7(\mathbf{x})$	− 3.45E + 00	− 3.41E + 00	− 3.42E + 00	− 3.42E + 00	− 3.43E + 00	− 3.43E + 00	− 3.43E + 00	− 3.2751
$f(\mathbf{x})$	**1.697082867**	1.74830941	1.728024	1.7312	1.72624	1.730499	1.724852309	1.724851989

* infeasible constraint

Table 15 Comparison results for the pressure vessel design problem

	ALSO	GA (Coello 2000)	PSO (He and Wang 2007)	BAT (Gandomi et al. 2013)	GWO (Mirjalili et al. 2014)	WOA (Mirjalili and Lewis 2016)	CSA (Askarzadeh 2016)	LS-II (Camarena et al. 2018)
$x_1(T_s)$	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125	0.812500214
$x_2(T_h)$	0.4375	0.4375	0.4375	0.4375	***0.4345**	0.4375	0.4375	0.437500122
$x_3(R)$	42.098437123378	40.3239	42.091266	42.0984456	42.089181	42.0982699	42.09844539	42.09844537
$x_4(L)$	176.6367864059546	200	176.7465	176.6365958	176.758731	176.638998	176.6365986	176.636599
$g_1(\mathbf{x})$	− 1.635188e-07	− 3.42E-02	− 1.39E-04	**8.00E-11*	− 1.79E-04	− 3.39E-06	− 3.97E-09	− 2.1836e-07
$g_2(\mathbf{x})$	− 0.03588	− 5.28E-02	− 3.59E-02	− 3.59E-02	− 3.30E-02	− 3.59E-02	− 3.59E-02	− 41.6968
$g_3(\mathbf{x})$	− 0.476468	− 3.04E + 02	− 1.16E + 02	− 4.97E-05	− 4.06E + 01	− 1.25E + 00	− 8.72E-04	− 0.0020
$g_4(\mathbf{x})$	− 63.3632136	− 4.00E + 01	− 6.33E + 01	− 6.34E + 01	− 6.32E + 01	− 6.34E + 01	− 6.34E + 01	− 63.3634
$f(\mathbf{x})$	**6059.717367**	6288.7445	6061.0777	**6059.714335**	**6051.5639**	6059.741	**6059.714363**	**6059.71623**

* infeasible constraint

exploration, whereas the second part improves the exploitation. The proposed algorithm maintains a proper balance between the exploration and exploitation of a search space. Besides, it can be adjusted according to λ. 22 benchmark functions were employed in order to investigate the performance of the proposed algorithm in terms of local optima avoidance and convergence. The results showed that the ALSO algorithm provides competitive results compared to well-known metaheuristics algorithms such as the PSO, ABC, CS, BAT, GWO, WOA, HHO, BRO, and RSO. On the one hand, the results on the unimodal benchmark functions concluded the superior convergence of the ALSO. On the other hand, the local optima avoidance ability of the ALSO was confirmed by the results on multimodal benchmark functions. Moreover, the ALSO algorithm was benchmarked on a very challenging CEC 2017. As per results, it can be seen that the ALSO was capable of finding excellent solutions compared to other algorithms. Finally, the proposed algorithm was applied to the three engineering design problems whose results prove that ALSO has high performance in the unknown, challenging search spaces. Besides, the runtime of the ALSO is less than the other algorithms. In addition to executing less runtime than other algorithms, the ALSO requires less memory space in high-dimensional problems. In the meantime, the methods in the literature cluster around the obtained optimal solution by focusing on it during the entire iteration and try to improve the sensitivity of the solution, whereas the ALSO does not give up the chance to reach a better solution during the entire iteration. To sum up, it can be pointed out that the proposed ALSO algorithm can be an alternative optimization algorithm to the methods in the literature.

Appendix

See to Tables 16 , 17 and 18

Table 16 Pair-wise statistical comparison between the ALSO and all compared algorithms by Wilcoxon rank-sum test for $D = 30$

	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h
F1	1.00e + 00: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.00e + 00: 0	1.00e + 00: 0	1.00e + 00: 0	1.00e + 00: 0	1.00e + 00: 0
F2	1.00e + 00: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.00e + 00: 0	3.47e-21: 1 +
F3	3.47e-21: 1-	3.47e-21: 1-	3.02e-04: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.31e-19: 1-	3.47e-21: 1 +
F4	8.95e-03: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F5	2.08e-01: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F6	4.31e-01: 0	2.18e-09: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.00e + 00: 0	3.47e-21: 1 +
F7	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	8.29e-17: 1 +	3.47e-21: 1-
F8	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.00e + 00: 0	1.00e + 00: 0	1.00e + 00: 0	3.47e-21: 1 +	1.00e + 00: 0
F9	1.94e-17: 1 +	6.69e-20: 1 +	1.09e-08: 1 +	5.02e-01: 0	4.68e-21: 1 +	8.55e-18: 1 +	3.47e-21: 1-	4.76e-20: 1 +	3.47e-21: 1 +
F10	1.97e-01: 0	9.58e-12: 1-	4.59e-12: 1-	3.47e-21: 1 +	2.66e-16: 1-	4.81e-14: 1-	2.66e-16: 1-	4.41e-02: 1 +	2.66e-16: 1-
F11	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F12	1.54e-20: 1-	3.47e-21: 1 +	1.54e-20: 1-	1.54e-20: 1-	3.47e-21: 1 +	1.54e-20: 1-	1.54e-20: 1-	3.47e-21: 1 +	4.77e-09: 1 +
F13	1.27e-08: 1-	3.47e-21: 1 +	2.55e-07: 1 +	3.47e-21: 1 +	2.34e-07: 1-	3.47e-21: 1 +	5.72e-21: 1 +	5.00e-20: 1 +	3.47e-21: 1 +
F14	4.12e-02: 1 +	9.37e-01: 0	6.44e-01: 0	3.93e-16: 1 +	1.64e-03: 1 +	4.26e-14: 1 +	4.82e-11: 1 +	5.63e-03: 1 +	4.68e-21: 1 +
F15	1.03e-19: 1 +	3.47e-21: 1 +	1.54e-14: 1 +	3.47e-21: 1 +	9.79e-01: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F16	1.24e-08: 1-	8.79e-09: 1 +	1.14e-12: 1-	1.47e-20: 1 +	2.63e-12: 1-	9.23e-12: 1 +	7.65e-19: 1 +	1.96e-16: 1 +	3.47e-21: 1 +
F17	3.47e-21: 1 +	1.68e-05: 1 +	3.03e-16: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	2.70e-18: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F18	1.32e-16: 1 +	3.47e-21: 1 +	1.16e-13: 1 +	3.47e-21: 1 +	8.54e-16: 1 +	3.47e-21: 1 +	1.08e-19: 1 +	5.20e-17: 1 +	3.47e-21: 1 +
F19	2.01e-08: 1 +	1.93e-10: 1-	8.09e-05: 1-	3.47e-21: 1 +	9.50e-01: 0	3.47e-21: 1 +	4.46e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F20	3.47e-21: 1 +	3.47e-21: 1 +	6.67e-01: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	7.37e-20: 1 +	9.38e-20: 1 +	3.47e-21: 1 +
F21	7.33e-01: 0	4.46e-21: 1 +	1.02e-14: 1 +	3.47e-21: 1 +	1.92e-03: 1 +	3.47e-21: 1 +	2.92e-20: 1 +	7.33e-21: 1 +	3.47e-21: 1 +
F22	9.12e-14: 1 +	3.47e-21: 1 +	1.63e-09: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
Total	17	19	17	21	18	19	18	21	20

Table 17 Pair-wise statistical comparison between the ALSO and all compared algorithms by Wilcoxon rank-sum test for $D = 50$

	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h
F1	1.00e + 00: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.00e + 00: 0	1.00e + 00: 0	1.00e + 00: 0	3.47e-21: 1 +	1.00e + 00: 0
F2	1.00e + 00: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F3	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.06e-14: 1-	3.47e-21: 1 +
F4	2.66e-16: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F5	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F6	4.59e-03: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F7	6.76e-14: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-
F8	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	8.75e-01: 0	8.75e-01: 0	8.75e-01: 0	3.47e-21: 1 +	8.75e-01: 0
F9	1.29e-01: 0	3.47e-21: 1 +	5.32e-04: 1-	1.58e-09: 1-	2.40e-02: 1-	8.42e-03: 1-	3.47e-21: 1-	2.38e-01: 0	5.29e-01: 0
F10	1.35e-03: 1-	3.47e-21: 1 +	9.22e-05: 1-	3.47e-21: 1 +	2.99e-12: 1-	4.11e-11: 1-	1.38e-12: 1-	2.94e-01: 0	1.38e-12: 1-
F11	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.17e-18: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F12	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	8.62e-08: 1 +
F13	1.88e-06: 1-	3.47e-21: 1 +	2.92e-20: 1 +	3.47e-21: 1 +	5.54e-11: 1-	3.47e-21: 1 +	3.47e-21: 1 +	1.47e-20: 1 +	3.47e-21: 1 +
F14	3.01e-10: 1 +	3.47e-21: 1 +	1.62e-17: 1 +	3.83e-21: 1 +	1.20e-05: 1 +	3.47e-21: 1 +	5.44e-21: 1 +	1.93e-19: 1 +	3.47e-21: 1 +
F15	5.00e-19: 1 +	3.47e-21: 1 +	8.02e-19: 1 +	3.47e-21: 1 +	1.57e-10: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F16	1.43e-15: 1-	1.17e-18: 1-	1.17e-18: 1-	1.20e-20: 1 +	1.28e-18: 1-	2.70e-06: 1 +	5.44e-21: 1 +	8.64e-05: 1 +	3.47e-21: 1 +
F17	4.93e-18: 1 +	4.46e-21: 1 +	1.74e-12: 1-	3.47e-21: 1 +	3.65e-21: 1 +	6.01e-21: 1 +	1.15e-20: 1 +	3.65e-21: 1 +	3.47e-21: 1 +
F18	2.21e-07: 1 +	3.47e-21: 1 +	1.08e-01: 0	3.47e-21: 1 +	5.02e-01: 0	3.47e-21: 1 +	5.56e-16: 1 +	2.29e-20: 1 +	3.47e-21: 1 +
F19	9.93e-12: 1 +	4.62e-01: 0	1.01e-04: 1 +	3.47e-21: 1 +	7.53e-04: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F20	1.27e-20: 1 +	3.47e-21: 1 +	4.10e-18: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	5.24e-20: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F21	2.57e-02: 1 +	3.47e-21: 1 +	1.38e-19: 1 +	3.47e-21: 1 +	1.22e-07: 1 +	3.47e-21: 1 +	1.97e-20: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F22	4.46e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
Total	17	21	16	20	16	18	18	21	20

Table 18 Pair-wise statistical comparison between the ALSO and all compared algorithms by Wilcoxon rank-sum test for $D = 100$

	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h	p-value: h
F1	1.58e-08: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-
F2	2.28e-07: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	5.42e-14: 1-	3.47e-21: 1 +	3.47e-21: 1 +
F3	4.38e-13: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	4.24e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F4	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F5	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F6	2.54e-13: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.18e-07: 1-	3.47e-21: 1 +	3.47e-21: 1 +
F7	8.81e-19: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-
F8	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-
F9	3.06e-01: 0	3.47e-21: 1 +	1.77e-17: 1-	2.25e-18: 1-	1.77e-17: 1-	2.14e-18: 1-	3.47e-21: 1-	3.47e-21: 1 +	2.91e-17: 1-
F10	1.08e-10: 1 +	3.47e-21: 1 +	1.74e-14: 1 +	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-
F11	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.24e-10: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F12	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1 +	8.62e-08: 1 +
F13	5.86e-04: 1-	3.47e-21: 1 +	1.59e-19: 1 +	3.47e-21: 1 +	3.46e-15: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F14	2.37e-10: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	4.68e-21: 1 +	2.81e-15: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.65e-21: 1 +	3.47e-21: 1 +
F15	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	5.44e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F16	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	1.32e-01: 0	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F17	3.47e-21: 1 +	3.47e-21: 1 +	1.49e-05: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F18	1.77e-17: 1 +	3.47e-21: 1 +	2.66e-17: 1 +	3.47e-21: 1 +	2.93e-04: 1-	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F19	1.61e-14: 1 +	3.47e-21: 1 +	4.11e-12: 1 +	3.47e-21: 1 +	3.42e-19: 1 +	3.47e-21: 1 +	3.65e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F20	3.47e-21: 1 +	3.47e-21: 1 +	4.29e-04: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F21	5.91e-06: 1 +	3.47e-21: 1 +	1.04e-20: 1 +	3.47e-21: 1 +	4.97e-17: 1 +	3.47e-21: 1 +	3.65e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
F22	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +	3.47e-21: 1 +
Total	18	22	20	20	15	16	14	22	17

Authors' contributions All authors' contributions are equal.

Funding The authors declare that they have no funding.

Data availability The authors declare that they have all the data used in the simulations.

Code availability The authors declare that they have pseudo code in manuscript. Also, they have source code for proposed and compared algorithm.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

Ahmed H, Glasgow J (2012) Swarm intelligence: concepts Models and Applications. Kingston, Ontario, Canada

Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23:715–734

Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12

Awad NH, Ali MZ, Suganthan PN, Liang JJ, Qu BY (2016) Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization.

Camarena O, Cuevas E, Pérez-Cisneros M, Fausto F, González A, Valdivia A (2018) Ls-II: an improved locust search algorithm for solving optimization problems. Math Probl Eng. https://doi.org/10.1155/2018/4148975

Chen S (2009) Locust swarms - a new multi-optima search technique. In: Proceedings of the 2009 IEEE congress on evolutionary computation, pp 1745–1752

Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127

Collett M, Despland E, Simpson SJ, Krakauer DC (1998) Spatial scales of desert locust gregarization. Proc Natl Acad Sci USA 95:13052–13055

Cuevas E, González A, Zaldívar D, Pérez-Cisneros M (2015) An optimisation algorithm based on the behaviour of locust swarms. Int J Bioinspired Comput 7(6):402–407

Dhiman G, Garg M, Nagar A, Kumar V, Dehghani M (2021) A novel algorithm for global optimization: rat swarm optimizer. J Ambient Intell Humaniz Comput 12:8457–8482

Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In: Proceedings of the proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), pp 1470–1477

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the proceedings of the sixth international symposium on micro machine and human science, pp 39–43

Ernst UR, Van Hiel MB, Depuydt G, Boerjan B, De Loof A, Schoofs L (2015) Epigenetics and locust life phase transitions. J Exp Biol 218(1):88–99

Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110–111:151–166

Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH (2020) Marine predators algorithm: a nature-inspired metaheuristic. Expert Syst Appl 152:113377

Farshi TR (2021) Battle royale optimization algorithm. Neural Comput Appl 33:1139–1157

Fathollahi-Fard AM, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R (2020) Red deer algorithm (RDA): a new nature-inspired meta-heuristic. Soft Comput 24:14637–14665

Fister Jr I, Yang XS, Fister I, Brest J, Fister D (2013) A brief review of nature-inspired algorithms for optimization. arXiv preprint arXiv:1307.4186

Gandomi AH, Yang XS, Alavi AH, Talatahari S (2013) Bat algorithm for constrained optimization tasks. Neural Comput Appl 22:1239–1255

Geem ZW, Kim JH, Loganathan G (2001) A new heuristic optimization algorithm: harmony search. SIMULATION 76(2):60–68

Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S (2019) Emperor penguins colony: a new metaheuristic algorithm for optimization. Evol Intell 12:211–226

Hassanien AE, Emary E (2016) Swarm intelligence: principles, advances, and applications. CRC Press, Boca Raton, Florida

He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20(1):89–99

Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Michigan

Hoyle G (1958) The leap of the grasshopper. Sci Am 198(1):30–35

Inglis GD, Goettel MS, Erlandson MA, Weaver DK (2007) Grasshoppers and locusts field manual of techniques in invertebrate pathology. Springer, Dordrecht, pp 627–654

Jain M, Maurya S, Rani A, Singh V (2018) Owl search algorithm: a novel nature-inspired heuristic paradigm for global optimization. J Intell Fuzzy Syst 34(3):1573–1582

Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175

Kang L, Chen X, Zhou Y et al (2004) The analysis of large-scale gene expression correlated to the phase changes of the migratory locust. Proc Natl Acad Sci USA 101(51):17611–17615

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471

Kesemen O, Özkul E (2018) Solving cross-matching puzzles using intelligent genetic algorithms. Artif Intell Rev 49:211–225

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Lim TY (2014) Structured population genetic algorithms: a literature survey. Artif Intell Rev 41:385–399

Lynn N, Suganthan PN (2015) Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm Evol Comput 24:11–24

Martens D, Baesens B, Fawcett T (2011) Editorial survey: swarm intelligence for data mining. Mach Learn 82:1–42

Meetei KT (2014) A survey: swarm intelligence vs. genetic algorithm. Int J Sci Res 3(5):231–235

Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98

Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61

Orujpour M, Feizi-Derakhshi MR, Rahkar-Farshi T (2020) Multimodal forest optimization algorithm. Neural Comput Appl 32:6159–6173

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248

Salih SQ, Alsewari AA (2020) A new algorithm for normal and large-scale optimization problems: nomadic people optimizer. Neural Comput Appl 32:10359–10386

Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47

Scott J (2005) The locust jump: an integrated laboratory investigation. Adv Physiol Educ 29(1):21–26

Simpson SJ, Sword GA (2008) Locusts. Curr Biol 18(9):R364–R366

Simpson SJ, McCaffery AR, Hägele BF (1999) A behavioural analysis of phase change in the desert locust. Biol Rev 74(4):461–480

Simpson SJ, Sword GA, Lo N (2011) Polyphenism in Insects. Curr Biol 21(18):R738–R749

Srinivasan D, Seow TH (2003) Particle swarm inspired evolutionary algorithm (PS-EA) for multiobjective optimization problems. In: Proceedings of the The 2003 congress on evolutionary computation, CEC '03, pp 2292–2297

Tamura K, Yasuda K (2011) Spiral dynamics inspired optimization. J Adv Comput Intell Intell Inform (JACIII) 15(8):1116–1122

Topaz CM, Bernoff AJ, Logan S, Toolson W (2008) A model for rolling swarms of locusts. Eur Phys J Spec Top 157:93–109

Topaz CM, D'Orsogna MR, Edelstein-Keshet L, Bernoff AJ (2012a) Locust dynamics: behavioral phase change and swarming. Comput Biol 8(8):1–11

Topaz CM, D'Orsogna MR, Edelstein-Keshet L, Bernoff AJ (2012b) Locust dynamics: behavioral phase change and swarming. Plos Comput Biol 8(8):1–11

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: González JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N (eds) Nature inspired cooperative strategies for optimization (NICSO 2010) studies in computational intelligence. Springer, Berlin, Heidelberg, pp 65–74

Yang XS (2010) Firefly algorithm, levy flights and global optimization. In: Bramer M, Ellis R, Petridis M (eds) Research and development in intelligent systems XXVI. Springer, London, pp 209–218

Yang XS (2012) Flower pollination algorithm for global optimization. In: Durand-Lose J, Jonoska N (eds) Unconventional computation and natural computation UCNC 2012 lecture notes in computer science. Springer, Berlin, Heidelberg, pp 240–249

Yang XS (2014) Swarm intelligence based algorithms: a critical analysis. Evol Intell 7:17–28

Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings of the 2009 world congress on nature & biologically inspired computing (NaBIC), pp 210–214

Yapici H, Cetinkaya N (2019) A new meta-heuristic optimizer: pathfinder algorithm. Appl Soft Comput 78:545–568

Yeniay Ö (2005) Penalty function methods for constrained optimization with genetic algorithms. Math Comput Appl 10(1):45–56

Zhang Q, Wang R, Yang J, Ding K, Li Y, Hu J (2017) Collective decision optimization algorithm: a new heuristic optimization method. Neurocomputing 221:123–137