The Ant Lion Optimizer

Seyedali Mirjalili *

School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia
Queensland Institute of Business and Technology, Mt Gravatt, Brisbane, QLD 4122, Australia

ABSTRACT

This paper proposes a novel nature-inspired algorithm called Ant Lion Optimizer (ALO). The ALO algorithm mimics the hunting mechanism of antlions in nature. Five main steps of hunting prey such as the random walk of ants, building traps, entrapment of ants in traps, catching preys, and re-building traps are implemented. The proposed algorithm is benchmarked in three phases. Firstly, a set of 19 mathematical functions is employed to test different characteristics of ALO. Secondly, three classical engineering problems (three-bar truss design, cantilever beam design, and gear train design) are solved by ALO. Finally, the shapes of two ship propellers are optimized by ALO as challenging constrained real problems. In the first two test phases, the ALO algorithm is compared with a variety of algorithms in the literature. The results of the test functions prove that the proposed algorithm is able to provide very competitive results in terms of improved exploration, local optima avoidance, exploitation, and convergence. The ALO algorithm also finds superior optimal designs for the majority of classical engineering problems employed, showing that this algorithm has merits in solving constrained problems with diverse search spaces. The optimal shapes obtained for the ship propellers demonstrate the applicability of the proposed algorithm in solving real problems with unknown search spaces as well. Note that the source codes of the proposed ALO algorithm are publicly available at http://www.alimirjalili.com/ALO.html.

1. Introduction

In recent years metaheuristic algorithms have been used as primary techniques for obtaining the optimal solutions of real engineering design optimization problems [1–3]. Such algorithms mostly benefit from stochastic operators [4] that make them distinct from deterministic approaches. A deterministic algorithm [5–7] reliably determines the same answer for a given problem with a similar initial starting point. However, this behaviour results in local optima entrapment, which can be considered as a disadvantage for deterministic optimization techniques [8]. Local optima stagnation refers to the entrapment of an algorithm in local solutions and consequently failure in finding the true global optimum. Since real problems have extremely large numbers of local solutions, deterministic algorithms lose their reliability in finding the global optimum.

Stochastic optimization (metaheuristic) algorithms [9] refer to the family of algorithms with stochastic operators including evolutionary algorithms [10]. Randomness is the main characteristic of stochastic algorithms [11]. This means that they utilize random

operators when seeking for global optima in search spaces. Although the randomised nature of such techniques might make them unreliable in obtaining a similar solution in each run, they are able to avoid local solutions much easier than deterministic algorithms. The stochastic behaviour also results in obtaining different solutions for a given problem in each run [12].

Evolutionary algorithms search for the global optimum in a search space by creating one or more random solutions for a given problem [13]. This set is called the set of candidate solutions. The set of candidates is then improved iteratively until the satisfaction of a terminating condition. The improvement can be considered as finding a more accurate approximation of the global optimum than the initial random guesses. This mechanism brings evolutionary algorithms several intrinsic advantages: problem independency, derivation independency, local optima avoidance, and simplicity.

Problem and derivation independencies originate from the consideration of problems as a black box. Evolutionary algorithms only utilize the problem formulation for evaluating the set of candidate solutions. The main process of optimization is done completely independent from the problem and based on the provided inputs and received outputs. Therefore, the nature of the problem is not a concern, yet the representation is the key step when utilizing evolutionary algorithms. This is the same reason why evolutionary algorithms do not need to derivate the problem for obtaining its global optimum.

---

* Address: School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia.
E-mail address: seyedali.mirjalili@griffithuni.edu.au
URL: http://www.alimirjalili.com

As another advantage, local optima avoidance is high due the stochastic nature of evolutionary algorithms. If an evolutionary algorithm is trapped in a local optimum, stochastic operator lead to random changes in the solution and eventually escaping from the local optimum. Although there is no guarantee for resolving this issue completely, stochastic algorithms have much higher probability to escape from local optima compared to deterministic methods. A very accurate approximation of the global optimum is not guaranteed as well, but with running an evolutionary algorithm several times the probability of obtaining a better solution is increased.

Last but not least, the simplicity is another characteristic of evolutionary algorithms. Natural evolutionary concepts or collective behaviours are the main inspirations for the majority of algorithms in this field where they are so simple. In addition, evolutionary algorithms follow a general and common framework, in which a set of randomly created solutions is enhanced or evolved iteratively. What makes algorithms different in this field is the method of improving this set.

Some of the most popular algorithms in this field are: Genetic Algorithms (GA) [14,15], Particle Swarm Optimization (PSO) [16], Ant Colony Optimization (ACO) [17], Differential Evolution (DE) [18], Evolutionary Programming (EP) [19] [20]. Although these algorithms are able to solve many real and challenging problems, the so-called No Free Lunch theorem [21] allows researchers to propose new algorithms. According to this theorem, all algorithms perform equal when solving all optimization problems. Therefore, one algorithm can be very effective in solving one set of problems but in effective on a different set of problems. This is the foundation of many works in this field. Some of the recent algorithms are: Grey Wolf Optimizer (GWO) [22], Artificial Bee Colony (ABC) algorithm [23], Firefly Algorithm (FA) [24,25], Cuckoo Search (CS) algorithm [26,27], Cuckoo Optimization Algorithm (COA) [28], Gravitational Search Algorithm (GSA) [29], Charged System Search (CSS) [30–33], Magnetic Charged System Search [34,35], Ray Optimization (RO) [36–38] algorithm, Colliding Bodies Optimization (CBO) [39–44] algorithm, Hybrid Particle Swallow Swarm Optimization (HPSSO) [45], Democratic Particle Swarm Optimization (DPSO) [46,47], Dolphin Echolocation (DE) [48,49], and Chaotic Swarming of Particles (CSP) [50].

This paper also proposes a new algorithm called Ant Lion Optimizer (ALO) as an alternative approach for solving optimization problems. As its name implies, the ALO algorithm mimics the intelligent behaviour of antlions in hunting ants in nature. The rest of the paper is organized as follows:

Section 2 presents the main inspiration of this paper and proposes the ALO algorithm. Experimental results of the test functions are provided in Section 3. Sections 4 and 5 solve several real problems to demonstrate the applicability of the proposed algorithm. Finally, Section 6 concludes the work and discusses possible future research.

2. Ant Lion Optimizer

In this section the inspiration of the ALO algorithm is first presented. The mathematical model and the ALO algorithm are then discussed in details.

2.1. Inspiration

Antlions (doodlebugs) belong to the Myrmeleontidae family and Neuroptera order (net-winged insects). The lifecycle of antlions includes two main phases: larvae and adult. A natural total lifespan can take up to 3 years, which mostly occurs in larvae (only 3–5 weeks for adulthood). Antlions undergo metamorphosis in a cocoon to become adult. They mostly hunt in larvae and the adulthood period is for reproduction.

Their names originate from their unique hunting behaviour and their favourite prey. An antlion larvae digs a cone-shaped pit in sand by moving along a circular path and throwing out sands with its massive jaw [51,52]. Fig. 1(a) shows several cone-shaped pits with different sizes. After digging the trap, the larvae hides underneath the bottom of the cone (as a sit-and-wait predator [53]) and waits for insects (preferably ant) to be trapped in the pit [53] as illustrated in Fig. 1(b). The edge of the cone is sharp enough for insects to fall to the bottom of the trap easily. Once the antlion realizes that a prey is in the trap, it tries to catch it. However, insects usually are not caught immediately and try to escape from the trap. In this case, antlions intelligently throw sands towards to edge of the pit to slide the prey into the bottom of the pit. When a prey is caught into the jaw, it is pulled under the soil and consumed. After consuming the prey, antlions throw the leftovers outside the pit and amend the pit for the next hunt.

Another interesting behaviour that has been observed in life style of antlions is the relevancy of the size of the trap and two things: level of hunger and shape of the moon. Antlions tend to dig out larger traps as they become more hungry [54] and/or when the moon is full [55]. They have been evolved and adapted this way to improve their chance of survival. It also has been discovered that an antlion does not directly observe the shape of the moon to decide about the size of the trap, but it has an internal lunar clock to make such decisions [55].

The main inspiration of the ALO algorithm comes from the foraging behaviour of antlion's larvae. In the next subsection the behaviour of antlions and their prey in nature is first modelled mathematically. An optimization algorithm is then proposed based on the mathematical model.

2.2. Operators of the ALO algorithm

The ALO algorithm mimics interaction between antlions and ants in the trap. To model such interactions, ants are required to move over the search space, and antlions are allowed to hunt them and become fitter using traps. Since ants move stochastically in nature when searching for food, a random walk is chosen for modelling ants' movement as follows:

$$X(t) = [0, \; cumsum(2r(t_1)-1), \; cumsum(2r(t_2)-1), \ldots, \; cumsum(2r(t_n)-1)]$$
$$(2.1)$$

where $cumsum$ calculates the cumulative sum, n is the maximum number of iteration, t shows the step of random walk (iteration in this study), and $r(t)$ is a stochastic function defined as follows:

$$r(t) = \begin{cases} 1 & if \; rand > 0.5 \\ 0 & if \; rand \leqslant 0.5 \end{cases} \qquad (2.2)$$

where t shows the step of random walk (iteration in this study) and $rand$ is a random number generated with uniform distribution in the interval of $[0,1]$.

To have an image of this random walk, Fig. 2 is provided that illustrates three random walks over 500 iterations. This figure shows that the random walk utilized may fluctuate dramatically around the origin (red[1] curve), have increasing trend (black curve), or have descending behaviour (blue curve).

The position of ants are saved and utilized during optimization in the following matrix:

$$M_{Ant} = \begin{bmatrix} A_{1,1} & A_{1,2} & \ldots & \ldots & A_{1,d} \\ A_{2,1} & A_{2,2} & \ldots & \ldots & A_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n,1} & A_{n,2} & \ldots & \ldots & A_{n,d} \end{bmatrix} \qquad (2.3)$$

---

[1] For interpretation of color in Fig. 2, the reader is referred to the web version of this article.
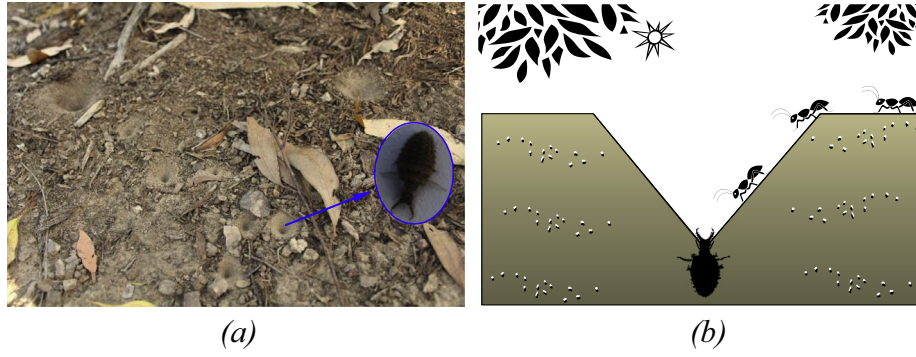
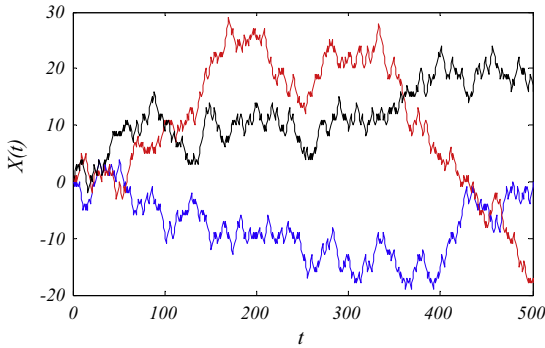Fig. 1. Cone-shaped traps and hunting behaviour of antlions.

Fig. 2. Three random walks.

where M_{Ant} is the matrix for saving the position of each ant, $A_{i,j}$ shows the value of the j-th variable (dimension) of i-th ant, n is the number of ants, and d is the number of variables.

It should be noted that ants are similar to particles in PSO or individuals in GA. The position of an ant refers the parameters for a particular solution. Matrix M_{Ant} has been considered to save the position of all ants (variables of all solutions) during optimization.

For evaluating each ant, a fitness (objective) function is utilized during optimization and the following matrix stores the fitness value of all ants:

$$M_{OA} = \begin{bmatrix} f([A_{1,1}, A_{1,2}, \ldots, A_{1,d}]) \\ f([A_{2,1}, A_{2,2}, \ldots, A_{2,d}]) \\ \vdots \\ \vdots \\ f([A_{n,1}, A_{n,2}, \ldots, A_{n,d}]) \end{bmatrix} \quad (2.4)$$

where M_{OA} is the matrix for saving the fitness of each ant, $A_{i,j}$ shows the value of j-th dimension of i-th ant, n is the number of ants, and f is the objective function.

In addition to ants, we assume the antlions are also hiding somewhere in the search space. In order save their positions and fitness values, the following matrices are utilized:

$$M_{Antlion} = \begin{bmatrix} AL_{1,1} & AL_{1,2} & \ldots & \ldots & AL_{1,d} \\ AL_{2,1} & AL_{2,2} & \ldots & \ldots & AL_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ AL_{n,1} & AL_{n,2} & \ldots & \ldots & AL_{n,d} \end{bmatrix} \quad (2.5)$$

where $M_{Antlion}$ is the matrix for saving the position of each antlion, $AL_{i,j}$ shows the j-th dimension's value of i-th antlion, n is the number of antlions, and d is the number of variables (dimension).

$$M_{OAL} = \begin{bmatrix} f([AL_{1,1}, AL_{1,2}, \ldots, AL_{1,d}]) \\ f([AL_{2,1}, AL_{2,2}, \ldots, AL_{2,d}]) \\ \vdots \\ \vdots \\ f([AL_{n,1}, AL_{n,2}, \ldots, AL_{n,d}]) \end{bmatrix} \quad (2.6)$$

where M_{OAL} is the matrix for saving the fitness of each antlion, $AL_{i,j}$ shows the j-th dimension's value of i-th antlion, n is the number of antlions, and f is the objective function.

During optimization, the following conditions are applied:

- Ants move around the search space using different random walks.
- Random walks are applied to all the dimension of ants.
- Random walks are affected by the traps of antlions.
- Antlions can build pits proportional to their fitness (the higher fitness, the larger pit).
- Antlions with larger pits have the higher probability to catch ants.
- Each ant can be caught by an antlion in each iteration and the elite (fittest antlion).
- The range of random walk is decreased adaptively to simulate sliding ants towards antlions.
- If an ant becomes fitter than an antlion, this means that it is caught and pulled under the sand by the antlion.
- An antlion repositions itself to the latest caught prey and builds a pit to improve its change of catching another prey after each hunt.

2.2.1. Random walks of ants

Random walks are all based on the Eq. (2.1). Ants update their positions with random walk at every step of optimization. Since every search space has a boundary (range of variable), however, Eq. (2.1) cannot be directly used for updating position of ants. In order to keep the random walks inside the search space, they are normalized using the following equation (min–max normalization):

$$X_i^t = \frac{(X_i^t - a_i) \times (d_i - c_i^t)}{(d_i^t - a_i)} + c_i \quad (2.7)$$

where a_i is the minimum of random walk of i-th variable, b_i is the maximum of random walk in i-th variable, c_i^t is the minimum of i-th variable at t-th iteration, and d_i^t indicates the maximum of i-th variable at t-th iteration.

Eq. (2.7) should be applied in each iteration to guarantee the occurrence of random walks inside the search space.

2.2.2. Trapping in antlion's pits

As discussed above, random walks of ants are affected by antlions' traps. In order to mathematically model this assumption, the following equations are proposed:

$$c_i^t = Antlion_j^t + c^t \qquad (2.8)$$

$$d_i^t = Antlion_j^t + d^t \qquad (2.9)$$

where c^t is the minimum of all variables at t-th iteration, d^t indicates the vector including the maximum of all variables at t-th iteration, c_j^t is the minimum of all variables for i-th ant, d_j^t is the maximum of all variables for i-th ant, and $Antlion_j^t$ shows the position of the selected j-th antlion at t-th iteration.

Eqs. (2.8) and (2.9) show that ants randomly walk in a hyper sphere defined by the vectors c and d around a selected antlion. A conceptual model of this behaviour is illustrated in Fig. 3.

Fig. 3 shows a two-dimensional search space. It may be observed that ants are required to move within a hypersphere around a selected antlion.

2.2.3. Building trap

In order to model the antlions's hunting capability, a roulette wheel is employed. As Fig. 3 show ants are assumed to be trapped in only one selected antlion. The ALO algorithm is required to utilize a roulette wheel operator for selecting antlions based of their fitness during optimization. This mechanism gives high chances to the fitter antlions for catching ants.

2.2.4. Sliding ants towards antlion

With the mechanisms proposed so far, antlions are able to build traps proportional to their fitness and ants are required to move randomly. However, antlions shoot sands outwards the center of the pit once they realize that an ant is in the trap. This behaviour slides down the trapped ant that is trying to escape. For mathematically modelling this behaviour, the radius of ants's random walks hyper-sphere is decreased adaptively. The following equations are proposed in this regard:

$$c^t = \frac{c^t}{I} \qquad (2.10)$$
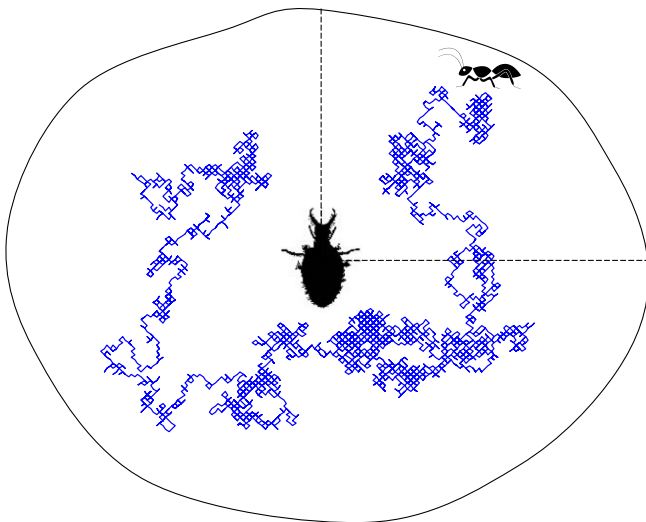
$$d^t = \frac{d^t}{I} \qquad (2.11)$$

where I is a ratio, c^t is the minimum of all variables at t-th iteration, and d^t indicates the vector including the maximum of all variables at t-th iteration.

In Eqs. (2.10) and (2.11), $I = 10^w \frac{t}{T}$ where t is the current iteration, T is the maximum number of iterations, and w is a constant defined based on the current iteration ($w = 2$ when $t > 0.1T$, $w = 3$ when $t > 0.5T$, $w = 4$ when $t > 0.75T$, $w = 5$ when $t > 0.9T$, and $w = 6$ when $t > 0.95T$). Basically, the constant w can adjust the accuracy level of exploitation.

Fig. 4 also shows the decreasing behaviour using Eqs. (2.10) and (2.11). These equations shrink the radius of updating ant's positions and mimics sliding process of ant inside the pits. This guarantees exploitation of search space.

2.2.5. Catching prey and re-building the pit

The final stage of hunt is when an ant reaches the bottom of the pit and is caught in the antlion's jaw. After this stage, the antlion pulls the ant inside the sand and consumes its body. For mimicking this process, it is assumed that catching prey occur when ants becomes fitter (goes inside sand) than its corresponding antlion. An antlion is then required to update its position to the latest position of the hunted ant to enhance its chance of catching new prey. The following equation is proposed in this regard:

$$Antlion_j^t = Ant_i^t \quad if\ f(Ant_i^t) > f\left(Antlion_j^t\right) \qquad (2.12)$$

where t shows the current iteration, $Antlion_j^t$ shows the position of selected j-th antlion at t-th iteration, and Ant_i^t indicates the position of i-th ant at t-th iteration.

2.2.6. Elitism

Elitism is an important characteristic of evolutionary algorithms that allows them to maintain the best solution(s) obtained at any stage of optimization process. In this study the best antlion obtained so far in each iteration is saved and considered as an elite. Since the elite is the fittest antlion, it should be able to affect the movements of all the ants during iterations. Therefore, it is assumed that every ant randomly walks around a selected antlion by the roulette wheel and the elite simultaneously as follows:

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \qquad (2.13)$$

where R_A^t is the random walk around the antlion selected by the roulette wheel at t-th iteration, R_E^t is the random walk around the elite at t-th iteration, and Ant_i^t indicates the position of i-th ant at t-th iteration.

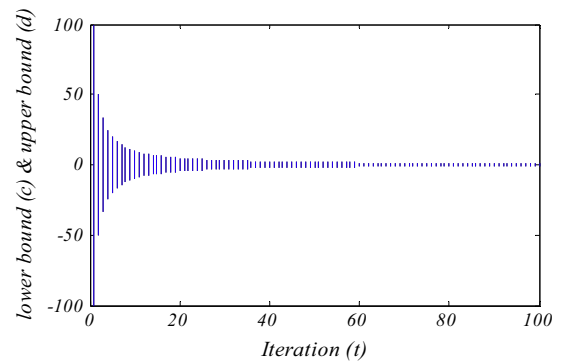**Fig. 3.** Random walk of an ant inside an antlion's trap.

Fig. 4. Adaptive lower (c^t) and upper (d^t) bounds.

2.3. ALO algorithm

With the proposed operators in the preceding subsections, the ALO optimization algorithm can now be defined. The ALO algorithm is defined as a three-tuple function that approximates the global optimum for optimization problems as follows:

$$ALO(A, B, C) \tag{2.14}$$

where A is a function that generates the random initial solutions, B manipulates the initial population provided by the function A, and C returns true when the end criterion is satisfied. The functions A, B, and C are defined as follows:

$$\emptyset \xrightarrow{A} \{M_{Ant}, M_{OA}, M_{Antlion}, M_{OAL}\} \tag{2.15}$$

$$\{M_{Ant}, M_{Antlion}\} \xrightarrow{B} \{M_{Ant}, M_{Antlion}\} \tag{2.16}$$

$$\{M_{Ant}, M_{Antlion}\} \xrightarrow{C} \{true, false\} \tag{2.17}$$

where M_{Ant} is the matrix of the position of ants, $M_{Antlion}$ includes the position of antlions, M_{OA} contains the corresponding fitness of ants, and M_{OAL} has the fitness of antlions.

The pseudo codes the ALO algorithm are defined as follows:

Initialize the first population of ants and antlions randomly
Calculate the fitness of ants and antlions
Find the best antlions and assume it as the elite (determined optimum)
while *the end criterion is not satisfied*
  for *every ant*
    Select an antlion using Roulette wheel
    Update c and d using equations Eqs. (2.10) and (2.11)
    Create a random walk and normalize it using Eqs. (2.1) and (2.7)
    Update the position of ant using (2.13)
  end for
  Calculate the fitness of all ants
  Replace an antlion with its corresponding ant it if becomes fitter (Eq. (2.12))
  Update elite if an antlion becomes fitter than the elite
end while
Return *elite*

The Matlab codes for the overall framework of the ALO algorithm as well as the functions A and B are provided in Appendix A.1–A.3.

In the ALO algorithm, the antlion and ant matrices are initialized randomly using the function A. In every iteration, the function B updates the position of each ant with respect to an antlion selected by the roulette wheel operator and the elite. The boundary of position updating is first defined proportional to the current number of iteration. The updating position is then accomplished by two random walks around the selected antlion and elite. When all the ants randomly walk, they are evaluated by the fitness function. If any of the ants become fitter than any other antlions, their positions are considered as the new positions for the antlions in the next iteration. The best antlion is compared to the best antlion found during optimization (elite) and substituted if it is necessary. These steps iterative until the function C returns false.

2.4. ALO software (toolbox)

The ALO algorithm is developed in Matlab and offered as an open-source optimization toolbox. The main user interface of this toolbox is illustrated in Fig. 5. The software designed allows users to define the number of search agents, maximum number of iterations, number of variables, upper bounds of variables, lower bounds of variables, and the name of the objective function easily. As shown in Fig. 5, these variables can be defined in the "parameters" section of the software. After defining the parameters, the problem starts to be optimized as soon as the user clicks on the "start optimization" button. The software then interactively draws the convergence curve (and history if chosen) and updates the obtained best optimum so far in the convergence curve section. Eventually, the information of the best obtained optimum is shown in the "final results" section. Note that the source codes of the ALO algorithm and the toolbox can be downloaded from http://www.alimirjalili.com/ALO.html or http://www.mathworks.com/matlab-central/profile/authors/2943818-seyedali-mirjalili.

2.5. Hypotheses about the ALO algorithm

Theoretically speaking, the proposed ALO algorithm is able to approximate the global optimum of optimization problems due to the following reasons:

- Exploration of the search space is guaranteed by the random selection of antlions and random walks of ants around them.
- Exploitation of search space is guaranteed by the adaptive shrinking boundaries of antlions' traps.
- There is high probability of resolving local optima stagnation due to the use of random walks and the roulette wheel.
- ALO is a population-based algorithm, so local optima avoidance is intrinsically high.
- Intensity of ants' movement is adaptively decreased over the course of iterations, which guarantees convergence of the ALO algorithm.
- Calculating random walks for every ant and every dimension promotes diversity in the population.
- Antlions relocate to the position of best ants during optimization, so promising areas of search spaces are saved.
- Antlions guide ants towards promising regions of the search space.
- The best antlion in each iteration is saved and compared to the best antlion obtained so far (elite).
- The ALO algorithm has very few parameters to adjust.
- The ALO algorithm is a gradient-free algorithm and considers problem as a black box.

In the next sections several test beds and real problems are employed to benchmark and confirm the performance of the ALO algorithm in solving optimization problems.

3. Results and discussion

In this section a number of test problems are selected to benchmark the performance of the proposed ALO algorithm. Three groups of test functions are employed with different characteristics to test the performance of the ALO algorithm from different perspectives. The test functions are divided the three groups: unimodal (Table 1), multi-modal (Table 2), and composite functions (Table 3) [20,56–58]. As their names imply, unimodal test functions have single optimum so they can benchmark the exploitation and convergence of an algorithm. In contrast, multi-modal test functions have more than one optimum, making them more challenging than unimodal functions. One of the optima is called global optimum and the rest are called local optima. An algorithm should avoid all the local optima to approach and determine the global optimum. Therefore, exploration and local optima avoidance of
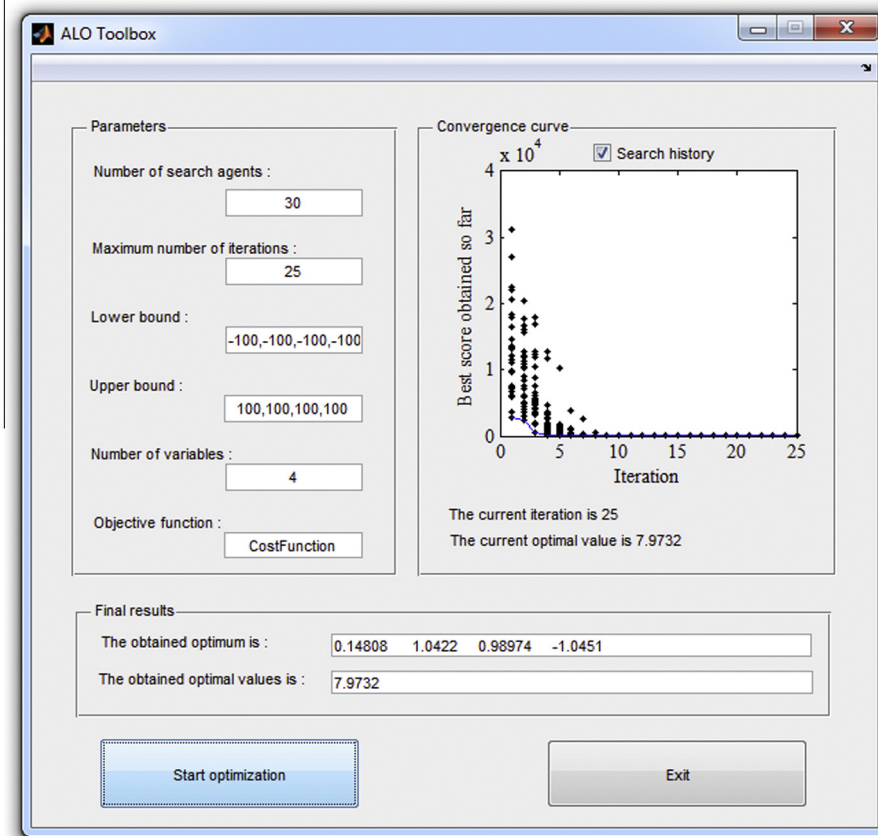
Fig. 5. User interface of the ALO toolbox (download it from http://www.alimirjalili.com/ALO.html).

Table 1
Unimodal benchmark functions.

Function	Dim	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30, 200	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$	30, 200	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30, 200	$[-100, 100]$	0
$F_4(x) = \max\{|x_i|, \ 1 \leqslant i \leqslant n\}$	30, 200	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30, 200	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^{n} (|x_i + 0.5|)^2$	30, 200	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30, 200	$[-1.28, 1.28]$	0

algorithms can be tested by the multi-modal test functions. Note that the minima of all the unimodal and multi-modal test functions are equal to 0 except the function F_8 where the minimum is changed based of the number of variables (*Dim*).

The last group of test functions, composite functions, are mostly the combined, rotated, shifted, and biased version of other uni-modal and multi-modal test functions [59,60]. As shown in Fig. 6, they mimic the difficulties of real search spaces by providing a massive number of local optima and different shapes for different regions of the search space. An algorithm should properly balance exploration and exploitation to approximate the global optima of such test functions. Therefore, exploration and exploitation combined can be benchmarked by this group of test functions.

For verification of the results of ALO, two well-known algorithms are employed: PSO [61] as the best algorithm among swarm-based technique and GA [62] as the best evolutionary algorithm. In addition the ALO algorithm is compared with several recent algorithms such as the States of Matter Search (SMS) algorithm [63,64], Bat

Algorithm (BA) [65], Flower Pollination Algorithm [66] , Cuckoo Search (CS) algorithm [26,27], and Firefly Algorithm (FA) [24,25]. In order to collect quantitative results, each algorithm is run on the test functions 30 times and the average and standard deviation of the best approximated solution in the last iteration is reported. These two metrics show us which algorithm behaves more stable when solving the test functions. Due to the stochastic nature of the algorithms, however, statistical test should also be conducted [67]. The averages and standard deviation only compare the overall performance of the algorithms, while a statistical test consider each run's results and prove that the results are statistically significant. The Wilcoxon rank-sum test [67,68] is conducted in this work.

The Wilcoxon rank-sum test is a non-parametric test in statics that can be used to determine/verify if two sets of solutions (population) are different statistically significant or not. In this method, each set of pairs in both populations are compared to calculate and analyse their differences. It also tests the null hypothesis as to whether both populations are of the same distribution. Technically speaking, this statistical test returns a parameter called p-vales. A p-value determines the significance level of two algorithms. In this work an algorithm is statistically significant if and only if it results in a p-value less than 0.05.

After all, each of the test functions is solved using 30 candidate solutions (antlions) over 1000 iterations and the results in Tables 4–9. Note that the average value of the minimums (*ave*) and their standard deviations (*std*) in 30 runs are provided in the table of results.

3.1. Results of the algorithms on unimodal test functions

According to the results of the algorithms on the unimodal test functions in Table 4, it is evident that the ALO algorithm

Table 2
Multimodal benchmark functions.

Function	Dim	Range	f_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	30, 200	$[-500, 500]$	$-418.9829 \times Dim$[a]
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30, 200	$[-5.12, 5.12]$	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30, 200	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30, 200	$[-600, 600]$	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30, 200	$[-50, 50]$	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30, 200	$[-50, 50]$	0

[a] *Dim* in F_8 indicates the number of variables.

Table 3
Composite benchmark functions (mathematical formulation of Sphere, Ackley, Rastrigin, Weierstrass, and Griewank can be found in Appendix A.4).

Function	Dim	Range	f_{min}
F_{14} (CF1): $f_1, f_2, f_3, \ldots, f_{10} = $ Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$	10	$[-5, 5]$	0
F_{15} (CF2): $f_1, f_2, f_3, \ldots, f_{10} = $ Griewank's Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$	10	$[-5, 5]$	0
F_{16} (CF3): $f_1, f_2, f_3, \ldots, f_{10} = $ Griewank's Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1, 1, 1, \ldots, 1]$	10	$[-5, 5]$	0
f_{17} (CF4): $f_1, f_2 = $ Ackley's Function $f_3, f_4 = $ Rastrigin's Function $f_5, f_6 = $ Weierstrass Function $f_7, f_8 = $ Griewank's Function $f_9, f_{10} = $ Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	10	$[-5, 5]$	0
f_{18} (CF5): $f_1, f_2 = $ Rastrigin's Function $f_3, f_4 = $ Weierstrass Function $f_5, f_6 = $ Griewank's Function $f_7, f_8 = $ Ackley's Function $f_9, f_{10} = $ Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	10	$[-5, 5]$	0
f_{19} (CF6): $f_1, f_2 = $ Rastrigin's Function $f_3, f_4 = $ Weierstrass Function $f_5, f_6 = $ Griewank's Function $f_7, f_8 = $ Ackley's Function $f_9, f_{10} = $ Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$	10	$[-5, 5]$	0

outperforms other algorithms on the majority of the test cases. The p-values in Table 5 also show that this superiority is statistically significant since the p-values are much less than 0.05. Considering the characteristic of unimodal test functions, it can be stated that the ALO algorithm benefits from high exploitation. High exploitation assists the ALO algorithm to rapidly converge towards the optimum and exploit it accurately as can also be inferred from the convergence curves in Fig. 7.

3.2. Results of the algorithms on multi-modal test functions

The results of the algorithms on multi-modal test functions are presented in Tables 6 and 7. Table 6 shows that again the ALO algorithm provides the best results on four of the test functions. The second best results belong the PSO and CS algorithms. The p-values reported in Table 7 also show that the ALO algorithm shows significantly better results. Considering the characteristics of
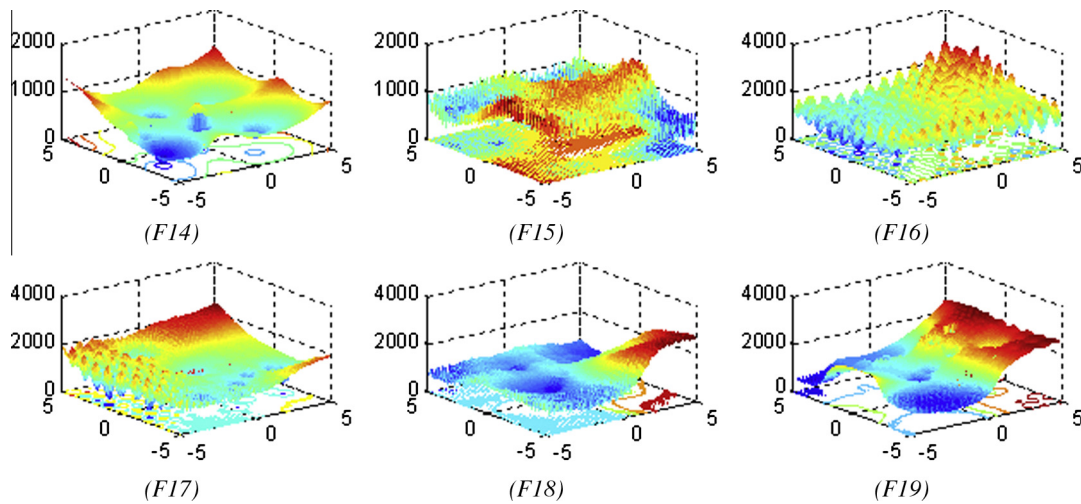
Fig. 6. Search space of composite benchmark functions.

Table 4
Results of unimodal benchmark functions (30-dimensional).

F	ALO		PSO		SMS		BA	
	ave	std	ave	std	ave	std	ave	std
F1	2.59E−10	1.65E−10	2.70E−09	1.00E−09	0.056987	0.014689	0.773622	0.528134
F2	1.84241E−06	6.58E−07	7.15E−05	2.26E−05	0.006848	0.001577	0.334583	3.816022
F3	6.06847E−10	6.34E−10	4.71E−06	1.49E−06	0.959865	0.82345	0.115303	0.766036
F4	1.36061E−08	1.81E−09	3.25E−07	1.02E−08	0.276594	0.005738	0.192185	0.890266
F5	0.346772393	0.109584	0.123401	0.216251	0.085348	0.140149	0.334077	0.300037
F6	2.56183E−10	1.09E−10	5.23E−07	2.74E−06	0.125323	0.084998	0.778849	0.67392
F7	0.004292492	0.005089	0.001398	0.001269	0.000304	0.000258	0.137483	0.112671
	FPA		CS		FA		GA	
	ave	std	ave	std	ave	std	ave	std
F1	1.06346E−07	1.27E−07	6.50E−03	2.05E−04	0.039615	0.01449	0.118842	0.125606
F2	0.000624246	0.000176	2.12E−01	3.98E−02	0.050346	0.012348	0.145224	0.053227
F3	5.6682E−08	3.9E−08	2.47E−01	2.14E−02	0.049273	0.019409	0.13902	0.121161
F4	0.003837885	0.002186	1.12E−05	8.25E−06	0.145513	0.031171	0.157951	0.862029
F5	0.781200043	0.366891	0.007197	0.007222	2.175892	1.447251	0.714157	0.972711
F6	1.08459E−07	1.25E−07	5.95E−05	1.08E−06	0.05873	0.014477	0.167918	0.868638
F7	0.003105276	0.001367	0.001321	0.000728	0.000853	0.000504	0.010073	0.003263

Table 5
p-Values of the Wilcoxon ranksum test over unimodal benchmark functions.

F	ALO	PSO	SMS	BA	FPA	CS	FA	GA
F1	N/A	0.025748	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F2	N/A	0.011330	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F3	N/A	0.000183	6.39e−05	0.000183	0.000183	0.000183	0.007285	0.000183
F4	N/A	0.005795	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F5	0.000183	0.000183	0.028571	0.000183	0.000183	N/A	0.003611	0.000183
F6	N/A	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F7	0.028571	0.000183	N/A	0.000183	0.000183	0.03738	0.73373	0.000183

multi-modal test functions and these results, it may be stated that the ALO algorithm has a high level of exploration which assists it to explore the promising regions of the search space. In addition, the local optima avoidance of this algorithm in satisfactory since it is able to avoid all of the local optima and approach the global optima on the majority of the multi-modal test functions. The convergence curves of the algorithms on some of the multi-modal test functions are illustrated in Fig. 8. This figure shows that the ALO shows the fastest convergence on multi-modal test functions as well.

3.3. Results of the algorithms on composite test functions

Tables 8 and 9 include the results of composite test functions. As per the results presented in Table 8, the ALO algorithm outperforms others in all of the composite test functions. The *p*-values reported in Table 9 show that this superiority is statistically significant on five of them (F15 to F19). The discrepancy of the results is high, which is due to the difficulty of the composite test functions that make them challenging for algorithms employed in this work.

Table 6
Results of multimodal benchmark functions (30-dimensional).

F	ALO		PSO		SMS		BA	
	ave	std	ave	std	ave	std	ave	std
F8	−1606.27643	314.4302	−1367.01	146.4089	−4.20735	9.36E−16	−1065.88	858.498
F9	7.71411E−06	8.45E−06	0.278588	0.218991	1.32512	0.326239	1.233748	0.686447
F10	3.73035E−15	1.5E−15	1.11E−09	2.39E−11	8.88E−06	8.56E−09	0.129359	0.043251
F11	0.018604494	0.009545	0.273674	0.204348	0.70609	0.907954	1.451575	0.570309
F12	9.74645E−12	9.33E−12	9.42E−09	2.31E−10	0.12334	0.040898	0.395977	0.993325
F13	2.00222E−11	1.13E−11	1.35E−07	2.88E−08	1.35E−02	2.88E−04	0.386631	0.121986
	FPA		CS		FA		GA	
	ave	std	ave	std	ave	std	ave	std
F8	−1842.42621	50.42824	−2094.91	0.007616	−1245.59	353.2667	−2091.64	2.47235
F9	0.273294621	0.068583	0.127328	0.002655	0.263458	0.182824	0.659271	0.815751
F10	0.007398721	0.007096	8.16E−09	1.63E−08	0.168306	0.050796	0.956111	0.807701
F11	0.085021659	0.040046	0.122678	0.049673	0.099815	0.024466	0.487809	0.217782
F12	0.000265711	0.000553	5.60E−09	1.58E−10	0.126076	0.263201	0.110769	0.002152
F13	3.67E−06	3.51E−06	4.88E−06	6.09E−07	0.00213	0.001238	1.29E−01	0.068851

Table 7
p-Values of the Wilcoxon ranksum test over multimodal benchmark functions.

F	ALO	PSO	SMS	BA	FPA	CS	FA	GA
F8	6.39e−05	6.39e−05	1.59e−05	6.39e−05	6.39e−05	N/A	6.39e−05	6.39e−05
F9	N/A	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F10	N/A	0.000246	0.000183	0.000183	0.000183	0.000246	0.000330	0.000183
F11	N/A	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183
F12	N/A	0.001421	0.000183	0.000183	0.000183	0.001421	0.000183	0.000183
F13	N/A	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183	0.000183

Table 8
Results of composite benchmark functions.

F	ALO		PSO		SMS		BA	
	ave	std	ave	std	ave	std	ave	std
F14	0.000151	0.000382	120	131.6561	776.4849	5.21e−12	182.476	117.0248
F15	14.56498	32.22876	162.9144	119.2351	873.7522	9.716179	487.2021	161.4107
F16	175.1532	46.50001	363.2361	151.3109	961.2754	67.21621	588.1938	137.7861
F17	316.0686	13.02047	450.0688	157.8496	899.8578	1.99e−05	756.9757	160.097
F18	4.399206	1.66107	175.3359	175.5385	740.9656	0.7858	542.2006	220.2014
F19	500.3161	0.206522	901.591	0.838585	900.4848	0.8442	818.5043	152.501
	FPA		CS		FA		GA	
	ave	std	ave	std	ave	std	ave	std
F14	0.337416	0.23641	110	110.0505	150.1696	97.15906	114.6139	26.96248
F15	18.23309	3.074685	140.6065	92.80327	314.4654	92.93417	95.46331	7.163383
F16	223.5667	50.25191	289.839	86.11561	734.5372	203.9693	325.4427	51.66827
F17	362.0262	54.01816	401.5247	98.16459	818.5732	109.9663	466.3074	29.56841
F18	10.1592	1.393908	212.7639	205.9728	133.5203	215.6027	90.36913	13.72977
F19	503.959	1.159453	812.2943	191.5134	862.2151	125.9599	521.1935	27.98507

Table 9
p-Values of the Wilcoxon ranksum test over composite benchmark functions.

F	ALO	PSO	SMS	BA	FPA	CS	FA	GA
F14	N/A	0.469654	0.000181	0.000183	0.000183	0.472676	0.000183	0.000183
F15	N/A	0.005795	0.000183	0.000183	0.002827	0.00044	0.000183	0.002202
F16	N/A	0.002202	0.000163	0.000183	0.037635	0.001315	0.000183	0.000246
F17	N/A	0.045155	6.39e−05	0.000183	0.002827	0.017257	0.000183	0.000183
F18	N/A	0.005795	6.39e−05	0.000183	0.000183	0.002827	0.000583	0.000183
F19	N/A	0.000183	6.39e−05	0.000183	0.000183	0.002827	0.000183	0.002827

The convergence curves of the algorithms on some of the composite test functions are illustrated in Fig. 9. This figure shows that the ALO shows the fastest convergence on the composite test functions as well.

Composite test functions benchmark the exploration and exploitation combined. Therefore, these results prove that the operators of the ALO algorithm appropriately balance exploration and exploitation to handle diverse difficulties in a challenging
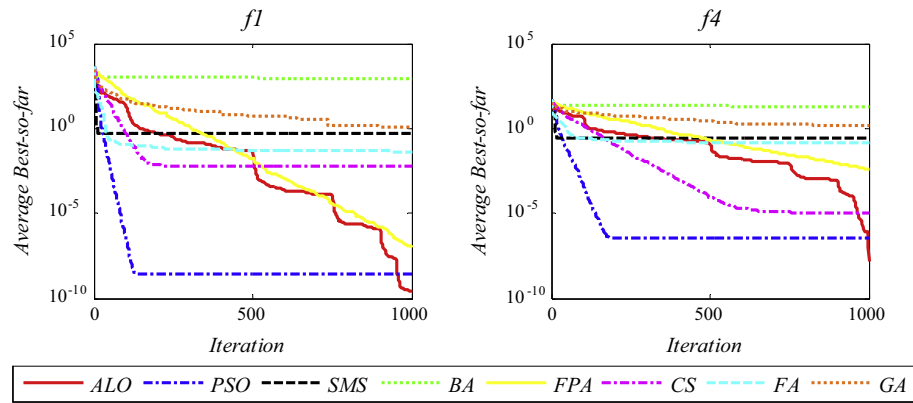
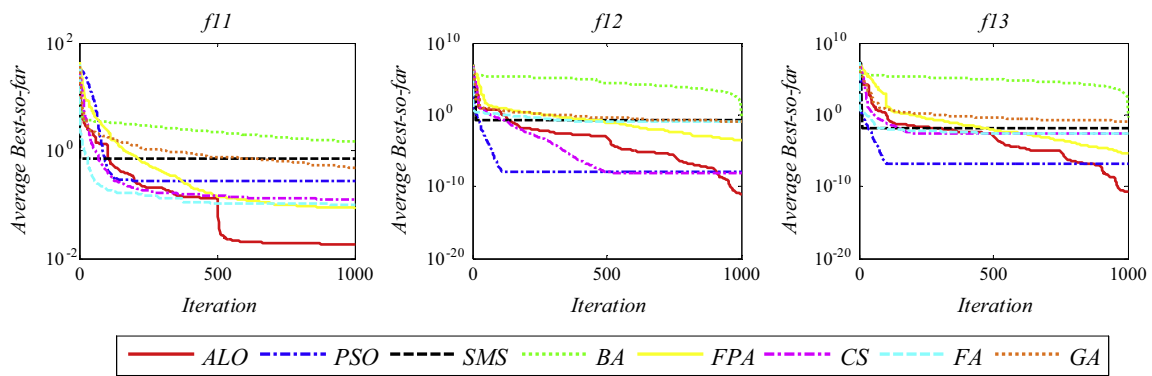**Fig. 7.** Convergence of algorithms on two of the unimodal test functions.

Fig. 8. Convergence of algorithms on three of the multi-modal test functions.

search space. Since the composite search spaces are highly similar to the real search spaces, these results make the ALO algorithm potentially able to solve challenging optimization problems.

3.4. Analysis of the ALO algorithm

For further observing the performance of the proposed ALO algorithm, five new metrics are employed in this subsection. In fact, this subsection aims for identifying and confirming the convergence and potential behaviour of the ALO algorithm when solving real problems. The employed quantitative metrics are as follows:

- The position of antlions from the first to the last iteration (search history).
- The interval of random walks.

- The position of antlions from the first to the last iteration (trajectory).
- The average fitness of antlions from the first to the last iterations.
- The fitness of the elite from the first to the last iteration (convergence).

Tracking the position of antlions during optimization allows observing how the ALO algorithm explores and exploits the search space. The interval of random walks shows how the ALO algorithm adaptively adjusts the boundary of random walks towards the optimum during optimization. The position of the antlions during optimization assists observing the movement of candidate solutions. There should be abrupt changes in the movements of candidate solutions in the exploration phase and gradual changes in the exploitation phase. The average fitness of antlions during
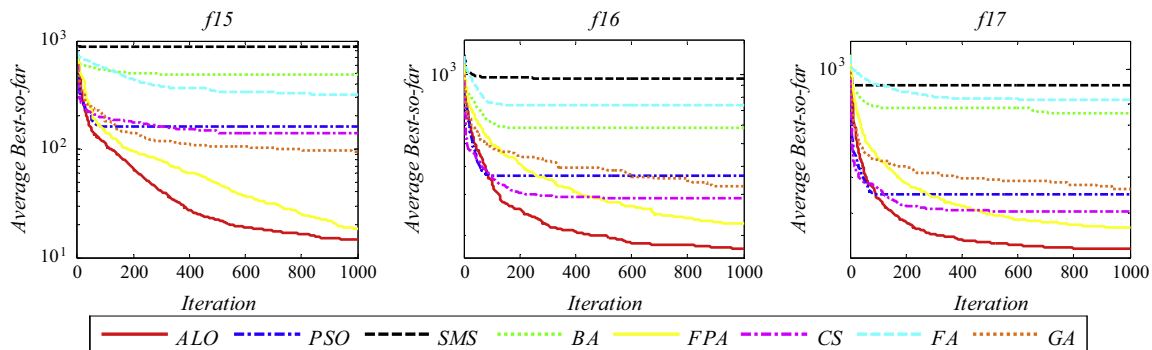

Fig. 9. Convergence of algorithms on three of the composite test functions.

optimization also shows the improvement in the fitness of the whole population during optimization. Finally, the fitness of the elite shows the improvement of the obtained global optimum during optimization.

Some of the functions are selected and solved by four search agents over 200 iterations. The results are illustrated in Fig. 10. The second column in Fig. 10 shows the history of antlion's position during optimization. It may be observed that the ALO algorithm tends to search the promising regions of the search space

extensively on all of the test functions. The behaviour of ALO when solving composite test functions is interesting, in which the coverage of search space seems to be high. In F17, for instance, two of the most promising areas close to the boundaries of the search space were explored, yet one of them was exploited eventually. This shows that the ALO's search agents are able to search the search space effectively.

The intervals of random walks in the third column in Fig. 10 show that the boundaries of random walks are adjusted towards
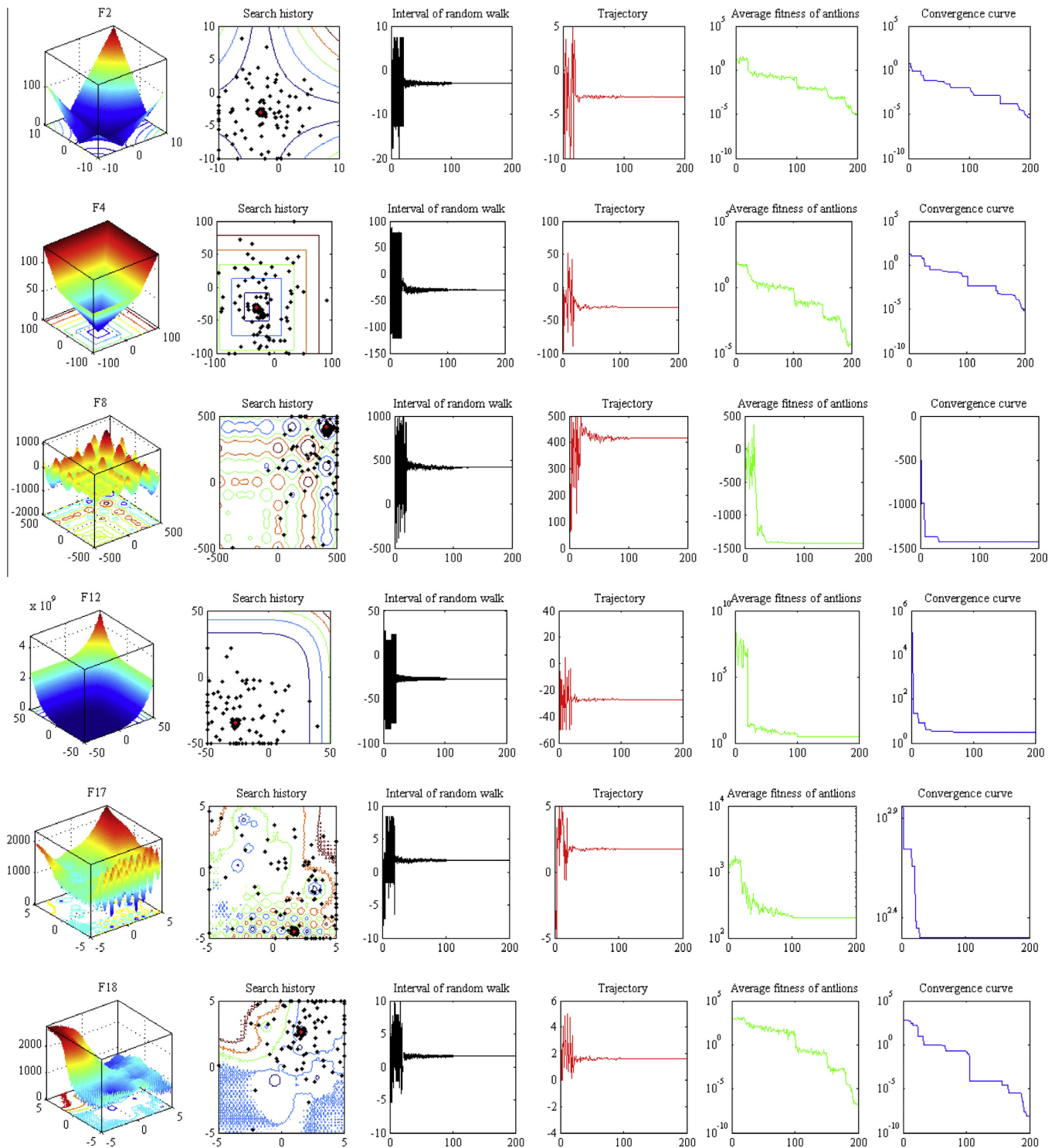

Fig. 10. Search history, trajectory in first dimension, average fitness of all antlions, and convergence rate.

the optimum over the course of iterations. It also may be observed that the boundaries are wide in the initial steps of iterations, while they are very narrow in the final steps of iterations. Another behaviour is the adaptive shrinking trend of boundaries which allows more accurate exploitation of the optimum as iteration increases.

The fourth column in Fig. 10 illustrates the trajectory of the first variable of the first antlion over 200 iterations. It can be observed that there are abrupt changes in the initial iterations. These changes are decreased gradually over the course of iterations. According to Berg et al. [69], this behaviour can guarantee that an algorithm eventually convergences to a point and searches locally in a search space.

The last two columns of Fig. 10 show the average fitness of all antlions and the elite, respectively. The average fitness of antlions shows a descending behaviour on all of the test functions. This proves that the ALO algorithm improves the overall fitness of the initial random population. A similar behaviour can be observed in the convergence curves. This also evidences that the approximation of the global optimum becomes more accurate as iteration increases. Another fact that can be seen is the accelerated trend in the convergence curves. This is due to the emphasis on local search and exploitation as iteration increases, which highly accelerate the convergence towards the optimum in the final steps of iterations.

3.5. Optimization of large-scale problems using ALO

To further prove the merits of the proposed ALO algorithm, this subsection solves the 200-dimensional versions of the unimodal and multimodal test functions investigated in the preceding subsections. Hundred search agents (candidate solutions) are employed to solve these test problems over 5000 iterations to generate the results for this subsection. After all, the results are presented in Tables 10 and 11.

As per the results shown in Table 10, the ALO algorithm outperforms all the other algorithms on four of the unimodal test functions (F1, F3, F5, and F6). In addition, Table 11 shows that this algorithm provides the best results on half of the multi-modal test functions (F10, F12, and F13). For the rest of the unimodal and multi-modal test functions, the ALO is ranked as the second best after CS, FA, or PSO. Poor performances of the majority of algorithms in Tables 10 and 11 show that such large-scale test functions can be very challenging. These results highly evidence that the ALO algorithm can be very effective for solving large-scale problems as well.

As summary, the results of this section proves that the proposed ALO algorithm shows high exploration and exploitation. For one, the proposed random walk mechanism and random selection of antlions promote exploration, assist the ALO algorithm to avoid local optima, and resolve local optima stagnation when solving challenging problems. For another, adaptive shrinking boundaries of antlions' traps and elitism emphasize exploitation as iteration increases, which leads to a very accurate approximation of the global optimum.

All these characteristics require the ALO algorithm to solve real optimization problems potentially. However, all of the problems solved so far have known search spaces. To prove whether the ALO algorithm is able to solve real problems with unknown search spaces as well, four real engineering problems are solved in the following sections. The first three problems are classical engineering design problems called: cantilever beam design, two-bar truss design, and gear design problems. These three problems are chosen deliberately because of their diverse characteristics. The cantilever beam design problem is chosen due to its medium number of parameter. The three-bar truss problem is selected due to its high number of constraints. Finally, the gear train design problem is chosen because of its discrete variables. These three problems can benchmark the ALO algorithm in solving real problems with diverse characteristics.

The last case study is a problem in the field of ship propeller design. This problem is a real Computational Fluid Dynamics (CFD) problem that has 20 variables and a large number of constraints. Therefore, the performance of the ALO algorithm can be benchmarked effectively in solving a super challenging problem.

4. Constrained optimization using the ALO algorithm

Real systems (problems) are mostly constrained. There are two types of constraints involved in defining the feasibly of solutions during the design process: inequality and equality constraints. For optimizing constrained problems, a constraint handling method should be integrated to the optimizer. There are several methods of constraints handling in the literature: penalty functions, special operators, repair algorithms, separation of objectives and constraints, and hybrid methods [70]. Since finding a good constraints handling method for the proposed ALO algorithm is out of the scope of this work, the simplest method called death penalty is employed. In this method, search agents that violate any of the constraints with any level are treated as the same and penalized by assigning a large fitness value (small objective value in case of maximization). This method is very cheap and readily applicable for the ALO algorithm without algorithm modifications.

In the following subsections this method is used to solve the constrained engineering problems. It should be noted that the statistical results are not provided anymore since it was already proved that the ALO algorithm is able to outperform other algorithm in a statistically significant manner. In addition, the main objective of solving a real problem is to achieve the global optimum with the least possible computational cost. Therefore, this section only presents the best obtained design and the required maximum number of function evaluations to determine it during 10 runs as the results of the ALO and other algorithms.

4.1. Cantilever beam design problem

A cantilever beam includes five hollow elements with square-shaped cross-section. Fig. 11 shows that each element is defined by one variable while the thickness is constant, so there is a total of 5 structural parameters. It may be see in Fig. 11 that there is also a vertical load applied to the free end of the beam (node 6) and the right side of the beam (node 1) is rigidly supported [71]. The objective is to minimize the weight of the beam. There is also one vertical displacement constraint that should not be violated by the final optimal design. The problem formulation is as follows:

Consider	$\vec{x} = [x_1 x_2 x_3 x_4 x_5],$
Minimize	$f(\vec{x}) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5),$
Subject to	$g(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \leqslant 1,$
Variable range	$0.01 \leqslant x_1, x_2, x_3, x_4, x_5 \leqslant 100,$

This problem has been solved by ALO and compared with the literature in Table 12. It may be seen that the comparison is made between Method of Moving Asymptotes (MMA) [71], Generalized Convex Approximation (GCA_I) [71], GCA_II [71], CS [72], and Symbiotic Organisms Search (SOS) [72].

Table 12 shows that the ALO algorithm outperforms other algorithms. This shows the high performance of the ALO algorithm in approximating the global optimum for this problem. The maximum number of function evaluations in Table 12 also shows that the ALO algorithm determines the global optimum for this problem with less number of function evaluations than the SOS algorithm.

Table 10
Results of unimodal benchmark functions (200-dimensional).

F	ALO		PSO		SMS		BA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F1	7.89e−07	1.10e−07	23.799	11.721	1039.213	0.4243	1117.34	20731
F2	530.82	222.67	237.87	22.432	1832.44	0.0122	3842.82	468.28
F3	2331.4	507.18	4693.34	503.57	2034.88	0.3780	1090.75	475.06
F4	30.58	1.1446	40.111	0.5879	300.265	0.0023	65.6670	2.8293
F5	167.04	49.746	911.2342	95.245	3863.53	0.5329	1410.80	591.07
F6	7.60e−07	7.39e−08	43.421	14.206	2494.43	0.0003	51.2056	12.005
F7	0.050546	0.014407	17.321	4.0133	28.359	1.99e−05	2.4344	0.12756
	FPA		CS		FA		GA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F1	55. 989	32. 678	3.80e−05	1.85e−05	76.128	1.5744	227.75	186.56
F2	280.6	6.9384	400.10	0.8656	611.19	71.219	6322.6	1092.7
F3	24,219	8540	12,957	633.75	14,852	6418.4	11,206	3986.1
F4	37.689	2.4572	30.936	1.6877	2.736	0.54729	101.54	2.5321
F5	3150.7	1490.6	332.67	159.88	1321.7	114.76	964.49	748.76
F6	166.99	41.109	8.17e−05	4.55e−05	78.42	2.3405	482.56	278.61
F7	4.8391	1.5354	0.40131	0.008707	0.0273	0.00411	116.56	60.161

Table 11
Results of multimodal benchmark functions (200-dimensional).

F	ALO		PSO		SMS		BA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F8	−44,426	1442.5	−18,136	4962.4	−35,969	0.8765	−25,632	869.47
F9	613.89	66.795	748.58	24.301	480.01	0.2365	723.38	100.96
F10	2.3058	0.25542	15.183	0.57627	17.293	0.0974	18.159	0.067775
F11	0.007424	0.00651	3241.2	137.49	4801.5	0.8532	4937	268.42
F12	5.3982	0.59591	4.07e+05	4.77e+05	1.00e+08	1.99e−05	1.69e+09	4.28e+08
F13	0.13915	0.22199	1.24e+06	5.82e+05	1.00e+08	1.99e−05	2.25e+09	8.85e+08
	FPA		CS		FA		GA	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F8	−45,771	3097.8	−52,600	156.04	−39,753	649.69	−28,660	1011
F9	702.95	69.653	541.58	41.889	475.45	28.058	1645.8	37.155
F10	17.544	0.16684	17.654	2.982	2.4297	0.038545	20.361	0.14256
F11	180.74	36.084	0.001191	0.001148	1.7048	0.014301	3306.8	113.3
F12	4.37e+07	3.22e+07	1.00e+10	0.0045	23.426	0.55985	8.14e+09	9.54e+08
F13	9.87e+07	3.80e+07	1.00e+10	0.0568	2.8614	0.0568	1.38e+10	1.45e++09

In addition, it is observed the ALO algorithm is able to find a design with the optimal weight identical to that of CS with a lower number of function evaluation. However, this algorithm needs 14,000 function evaluations to outperform both SOS and CS algorithms.

4.2. Three-bar truss design problem

The second problem is to design a three-bar truss to minimize its weight [72,74]. The objective function is very simple, yet the problem is highly constrained. The structural design problems usually have a large number of constraints. The constraints here are stress, deflection, and buckling constraints. The mathematical formulation of this problem is as follows:

Consider $\vec{x} = [x_1\ x_2] = [A_1 A_2]$,

Minimize $f(\vec{x}) = (2\sqrt{2}x_1 + x_2) * l$,

Subject to

$g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1 x_2} P - \sigma \leqslant 0$,

$g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1 x_2} P - \sigma \leqslant 0$,

$g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leqslant 0$,

Variable range $0 \leqslant x_1, x_2 \leqslant 1$,

where $l = 100$ cm, $P = 2$ kN/cm^2, $\sigma = 2$ kN/cm^2

The overall structure of the truss is illustrated in Fig. 12.

The algorithms that are chosen for comparison are: Differential Evolution with Dynamic Stochastic selection (DEDS) [75], Hybrid Particle Swarm Optimization with Differential Evolution (PSO-DE) [76], Mine Blast Algorithm (MBA) [74], Tsa [77], and CS [72]. The comparison results are provided in Table 13.

Table 13 shows that the ALO algorithm provides very competitive results and its best solution obtained is ranked as the second best solution after that of PSO-DE. It also provides very close results compared to PSO-DE (discrepancy is equal to 0.0000001). This again shows that the ALO algorithm is able to solve real constrained problems effectively. It should be noted that the optimal weight of the design obtained by Tsa violates one of the constraints [72]. As per the maximum number of function evaluations reported in Table 13, the ALO algorithm needs only 14,000 function evaluation to find a design with the optimal weight of 263.8958434, which is less than other algorithms.

4.3. Gear train design problem

As shown in Fig. 13, this problem is a discrete case study with four parameters. The objective is to find the optimal number of tooth for four gears of a train for minimizing the gear ratio [79,80]. To handle
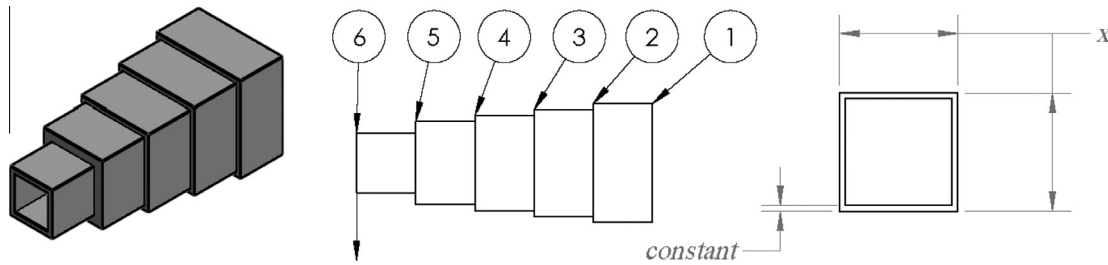
Fig. 11. Cantilever beam design problem.

Table 12
Comparison results for cantilever design problem.

| Algorithm	Optimal values for variables					Optimum weight	Max. eval.
	x_1	x_2	x_3	x_4	x_5		
ALO	6.01812	5.31142	4.48836	3.49751	2.158329	1.33995	14,000
SOS [73]	6.01878	5.30344	4.49587	3.49896	2.15564	1.33996	15,000
CS [72]	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999	2500
MMA [71]	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400	N/A
GCA_I [71]	6.0100	5.30400	4.4900	3.4980	2.1500	1.3400	N/A
GCA_II [71]	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400	N/A
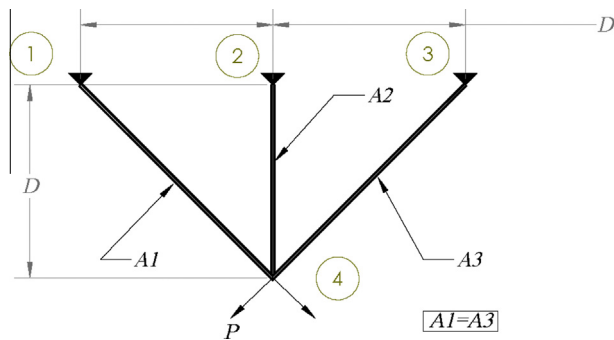
Fig. 12. Three-bar truss design problem.

discrete parameters, each search agent was rounded to the nearest integer number before the fitness evaluation.

The mathematical formulation of this problem is as follows:

Consider $\qquad \vec{x} = [x_1 x_2 x_3 x_4] = [n_A n_B n_C n_D],$

Minimize $\qquad f(\vec{x}) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2,$

Subject to $\qquad 12 \leqslant x_1, x_2, x_3, x_4 \leqslant 60,$

The best optimal design obtained by the ALO and other algorithms in the literature are presented in Table 14.

This table shows that the proposed ALO algorithm finds a design with the optimal value identical to those of Artificial Bee Colony

(ABC) algorithm [74], MBA [74], CS [72], and Interior Search Algorithm (ISA) [79]. However, the optimal values for variables obtained are different. Therefore, this design can be considered as a new design with a similar optimal gear ratio. However, the maximum numbers of function evaluations presented in Table 14 show that the ALO requires much less computational cost to determine the global optimum. Once more, these results prove that the proposed ALO algorithm can solve discrete real problems efficiently with low computational cost.

The results of these three engineering problems strongly evidence the merits of the ALO algorithm in solving problems with unknown search spaces. The results also show that the proposed algorithm is suitable for constrained and discrete problems. This is due to the employed mechanism of saving the best feasible solutions obtained as the elite and antlions, in which the ants are guided towards promising feasible regions of the search space. This leads to boosting the exploration of the feasible areas of a search space.

To further confirm and demonstrate these statements, two ship propellers are optimized by the proposed ALO algorithm in the next section.

5. Ship propeller design using ALO

Propellers provide thrust for moving marine and aero vehicles. Due to the high density of water, efficiency of a propeller becomes very important. The propellers that are going to be optimized by

Table 13
Comparison results of the three-bar truss design problem.

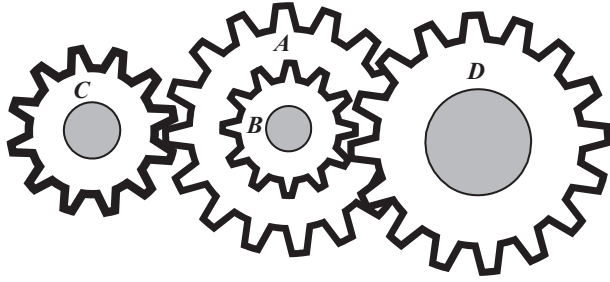| Algorithm	Optimal values for variables		Optimal weight	Max. eval.
	x_1	x_2		
ALO	0.788662816000317	0.408283133832901	263.8958434	14,000
DEDS [75]	0.78867513	0.40824828	263.8958434	15,000
PSO-DE [76]	0.7886751	0.4082482	263.8958433	17,600
MBA [74]	0.7885650	0.4085597	263.8958522	20,000
CS [72]	0.78867	0.40902	263.9716	15,000
Ray and Sain [78]	0.795	0.395	264.3	N/A
Tsa [77]	0.788	0.408	263.68 (infeasible)	N/A

Fig. 13. Gear train design problem.

Table 14
Comparison results of the gear train design problem.

Algorithm	Optimal values for variables				f_{min}	Max. eval.
	n_A	n_B	n_C	n_D		
ALO	49	19	16	43	2.7009e−012	120
CS [72]	43	16	19	49	2.7009e−012	5000
MBA [74]	43	16	19	49	2.7009e−012	10,000
ISA [79]	N/A	N/A	N/A	N/A	2.7009e−012	200
GA [81]	N/A	N/A	N/A	N/A	2.33e−07	10,000
ABC [82]	19	16	44	49	2.78e−11	40,000
GA [83]	33	14	17	50	1.362e−09	N/A
ALM [84]	33	15	13	41	2.1469e−08	N/A

the ALO algorithm as case studies have 2 m diameter and 4 or 5 blades. The 3D model of the five-blade propeller is illustrated in Fig. 14.

Achieving the highest efficiency is the main goal of propeller design which is defined as follows [85]:

$$\eta = \frac{V_a}{2\pi nD} \times \frac{K_T(x)}{K_Q(x)} \tag{5.1}$$

where V is axial velocity, D is the diameter of the propeller, n is the rotation speed of the propeller, K_T indicates the thrust coefficient, and K_Q shows that torque coefficient.

K_T is calculated as follows:

Table 15
Initial parameters, constant structural parameters, and operating conditions of the propeller.

Parameter	Value
Number of blades	4, 5
Diameter	2 m
Rotation speed	170 RPM
Ship speed	5 m/s (9.7192 knots)
Thrust	40,000 N
Torque	16183.1936 N m
Power	288099.0115 W
Density of water	999.97 kg/m³

$$K_T(x) = \frac{T}{\rho n^2 D^2} \tag{5.2}$$

where ρ shows the fluid density, T is the thrust, n indicates the rotation speed of the propeller, and D is the diameter length.

The structural parameters of a propeller are defined based on the representation method of the shape of the blades. A popular method for representing and modelling the shape of a propeller's blade is called Bézier curves. This method utilizes a set of controlling parameter to define the curvature of the airfoils along the blade. Another simple method is to define standard airfoils as the cross sections of a propeller's blade and adjust their special parameters. In this study, the second method is chosen due to its simplicity. Fig. 15 shows the structural parameters when using standard airfois.

We consider 10 airfoils along the blade, so the total number of parameters to be optimized by the ALO algorithm is 20. The parameters are as follows:

- $x_1 - x_2$: the chord length and maximum thickness of the first airfoil
- $x_3 - x_4$: the chord length and maximum thickness of the second airfoil
- . . .
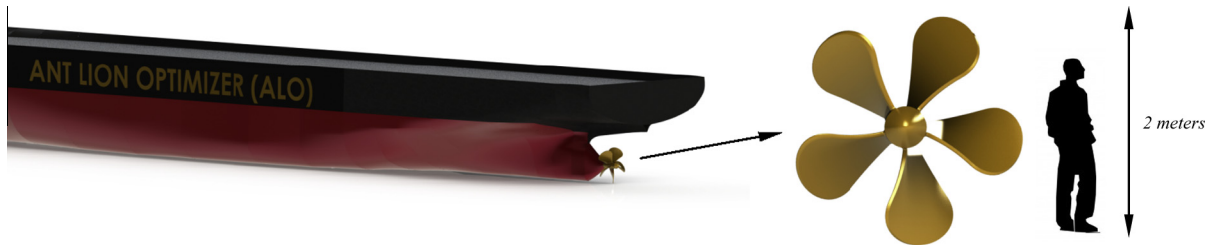- $x_{19} - x_{20}$: the chord length and maximum thickness of the tenth (last) airfoil
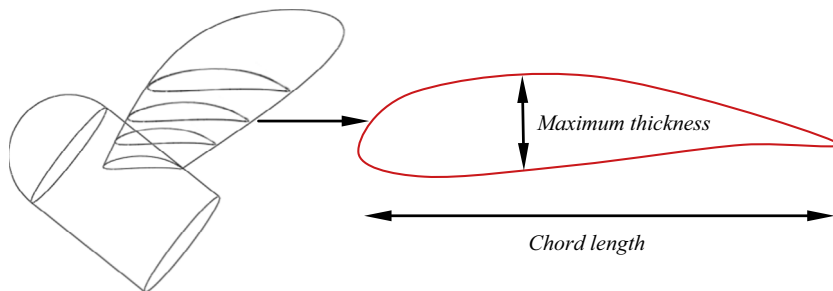

Fig. 14. Fixed pitch ship propeller with 5 blades.

Fig. 15. Airfoils along the blade define the shape of the propeller (NACA a = 0.8 meanline and NACA 65A010 thickness) [86].

Table 16
Best design parameters obtained using ALO for the four-blade propeller.

Chord1	Thickness1	Chord2	Thickness2	Chord3	Thickness3	Chord4	Thickness4	Chord5	Thickness5
0.1321	0.1421	0.1680	0.1852	0.2177	0.2252	0.2058	0.1699	0.1164	0.0012
Chord6	Thickness6	Chord7	Thickness7	Chord8	Thickness8	Chord9	Thickness9	Chord10	Thickness10
0.0300	0.0290	0.0211	0.0171	0.0150	0.0132	0.0096	0.0041	0.0034	0.000011

Table 17
Best design parameters obtained using ALO for the five-blade propeller.

Chord1	Thickness1	Chord2	Thickness2	Chord3	Thickness3	Chord4	Thickness4	Chord5	Thickness5
0.1330	0.1457	0.1691	0.1897	0.2200	0.2252	0.1955	0.1529	0.1130	0.0008
Chord6	Thickness6	Chord7	Thickness7	Chord8	Thickness8	Chord9	Thickness9	Chord10	Thickness10
0.0350	0.0316	0.0249	0.0152	0.0147	0.0143	0.0088	0.0041	0.0033	0.0000097

Table 18
Performance of the optimal design obtained.

Name	Four-blade Value	Five-blade Value
J	0.88235	0.88235
K_T	0.30382	0.30382
K_Q	0.06224	0.062153
Effy	0.68551	0.68647
AdEffy	0.82919	0.82919

The objective is also the maximization of efficiency. Therefore, this problem can be formulated as a maximization problem as follows:

$$\text{Suppose} : \vec{X}_i = (x_1, x_2, \ldots, x_n), \tag{5.3}$$

$$\text{Maximize} : \eta(x) \tag{5.4}$$
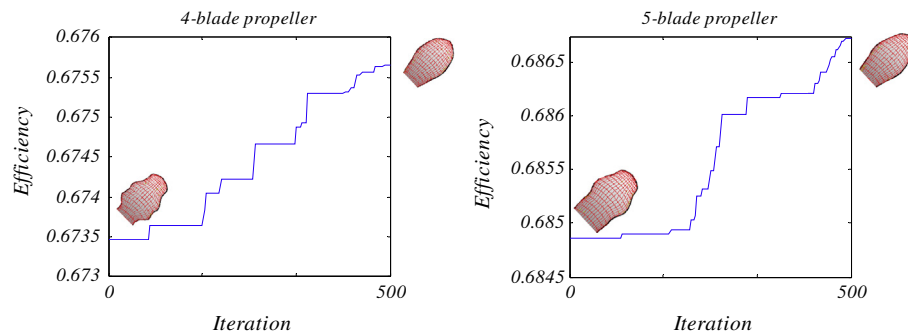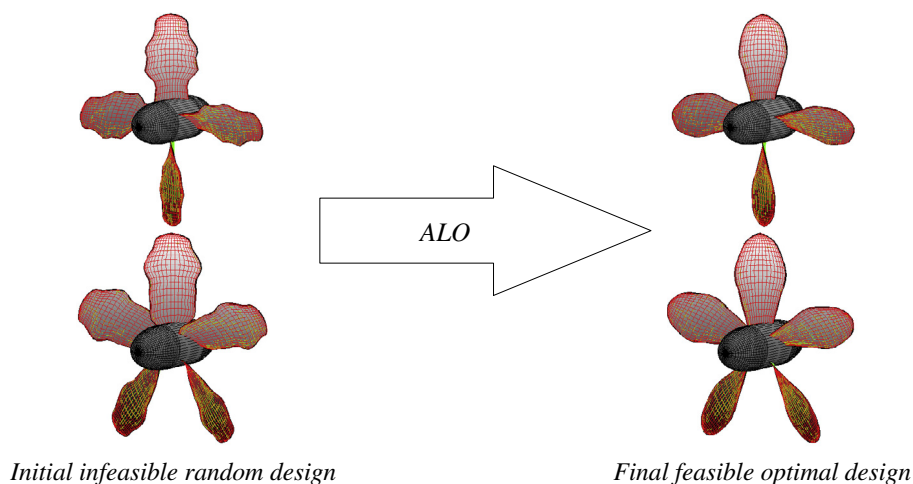
$$\text{Subject to} : \text{wake friction and thrust deduction} \tag{5.5}$$

Fig. 16. Convergence of the ALO algorithm when finding the optimal shapes for the four- and five-blade propellers.

Initial infeasible random design   *Final feasible optimal design*

Fig. 17. Improved design from initial infeasible design to feasible optimal design.

Table 19
Mathematical formulation of the primitive functions in Table 3.

Name	Formulation
Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$
Ackley	$f(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$
Griewank	$\frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Weierstrass	$\sum_{i=1}^{D}\left(\sum_{k=0}^{kmax}\left[a^k \cos(2\pi b^k(x_i + 0.5))\right]\right) - D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k 0.5)], \ a = 0.5, \ b = 3, \ kmax = 20$
Rastrigin	$\sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$

$$\text{Parameter range}: \ 0 < x_1 - x_{10} \leqslant 1 \qquad (5.6)$$

It should be noted that the full list of all constrains is not reported due the large number and complexity of their mathematical formulations. The large number of constraints makes this problem very challenging and expensive, in which each function evaluation can take up to 5 min for each design. As the simulator, an open source software called Openprop [87] is utilized. The initial parameters, constant structural parameters, and operating conditions of the propeller are listed in Table 15.

To approximate the global optimum of this problem using the ALO algorithm, 40 search agents and 500 iterations are employed. Due to the expensive computational cost of the objective function, the algorithm was run only twice, which took around a week. The best optimal design parameters obtained are provided in Tables 16–18. Note that two different propellers with four and five blades are considered.

The optimal efficiencies obtained for both propellers are 0.68551 and 0.68647. Other performance indicators of the optimal design obtained are presented in Table 18.

The fitness history of the elite when optimizing the shape of propeller is illustrated in Fig. 16, which show the improvements of the initial random shape during optimization.

For further observation of the impacts of ALO in improving the initial random shapes for the propellers, Fig. 17 is provided. This figure shows how the ALO algorithm finds a very smooth shape for the blades in order to maximize efficiency for both propellers.

Once more, all these results prove the merits of the proposed algorithm in solving challenging problems with unknown search spaces. Therefore, this powerful optimization technique is offered as a tool for finding the optimal designs of optimization problems in different fields of study.

6. Conclusion

This work proposed a novel nature-inspired algorithm called ALO. The hunting behaviour of antlions and entrapment of ants in antlions' traps were the main inspirations for this algorithm. Several operators were proposed and mathematically modelled for equipping the ALO algorithm with high exploration and exploitation. The performance of the proposed algorithm was benchmarked on 19 test functions in terms of exploration, exploitation, local optima avoidance, fitness improvement of the population, search history, trajectory of antlions, and convergence rate. As per the superior results of the ALO on the majority of the unimodal test functions and convergence curves, it can be concluded that the proposed algorithm benefits from high exploitation and convergence rate. The main reason for the high exploitation and convergence speed is due to the proposed adaptive boundary shrinking mechanism and elitism. High exploration of ALO can be concluded from the results of multimodal and composite test functions, which is due the employed random walk and roulette wheel selection mechanisms.

From the average fitness of all antlions during optimization, it is evidently concluded that the ALO algorithm successfully improves the overall fitness of random initial solutions on optimization problems. In addition, the convergence curves and search histories proved that the ALO algorithm effectively searches and converges towards the most promising regions of the search space. Therefore, the proposed algorithm has the potential to discover different regions of a given problem. Finally, the trajectories evidenced that the ALO algorithm requires antlions to move abruptly in the initial steps of optimizing and locally in the final steps of optimization, which leads to a smooth balance and transition between exploration and exploitation.

The ALO algorithm was compared to seven well-known and recent algorithms in the literature: PSO and GA, SMS, BA, FPA, CS, and FA. Wilcoxon statistical tests were also conducted when comparing the algorithms. The results showed that the proposed algorithm provides very competitive results and outperforms other algorithms in the majority of test functions. The statistical test also proved that the results were statistically significant for the ALO algorithm. Therefore, it may be concluded from the comparative results that the proposed ALO algorithm is able to be employed as an alternative optimizers for optimizing different problems.

The paper also considered solving three classical engineering problems and one challenging CFD problem using the ALO algorithm. The results of the ALO algorithm on these real problems were compared to a wide range of other algorithms in the literature. The comparative results demonstrated that the ALO algorithm is able to solve real problems with unknown search spaces as well. With these findings, another conclusion is that the proposed ALO algorithm is able to solve real problems with unknown search spaces as well. Other conclusion remarks that can be made from the results of this study are as follows:

- Random selection of antlions using a roulette wheel guarantees exploration of the search space.
- Random walks of ants around the antlions also emphasize exploration of the search space around the antlions.
- The use of random walk and roulette wheel assist the ALO algorithm to resolve local optima stagnations.
- Local optima avoidance is very high since the ALO algorithm employs a population of search agents to approximate the global optimum.
- Calculating random walks for every ant and every dimension causes diverse movement behaviours for ants inside the antlions' traps and maintains the diversity of position explored around antlions.
- Adaptive shrinking mechanism for defining the boundaries of random walks promotes exploitation as the iteration counter increases.
- Intensity of ants' movement is adaptively decreased over the course of iterations, which guarantee convergence of the ALO algorithm.

- Promising areas of search spaces are saved since antlions relocate to the position of the best ants during optimization.
- The best antlion in each iteration is saved and considered as the elite, so all ants tend towards the best solution obtained so far as well.
- The ALO algorithm has very few parameters to adjust of which some of them are adaptive, so it is a flexible algorithm for solving diverse problems.
- The ALO algorithm is a gradient-free algorithm and considers problems as a black box, so it is readily applicable for solving real problems.

Several research directions can be recommended for future studies with the proposed algorithm. Solving different optimization problems in different fields can be done. In addition, extending this algorithm to solve multi- and many-objective problems can be considered as a good contribution. Another research direction is to investigate the effectiveness of other random walks such as lévy flight in improving the performance of the ALO algorithm.

Appendix A

A.1. Matlab codes for the overall framework of the ALO

```
{M_Ant, M_Antlion} = A();
while C(M_Ant, M_Antlion) ≠ true
    {M_Ant, M_Antlion} = B(M_Ant, M_Antlion)
end while
```

A.2. Matlab codes for the function A

```
for i = 1: n
    for j = 1: d
        M_Ant (i,j) = (d(i) − c(i)) ∗ rand() + c(i);
        M_Antlion (i,j) = (d(i) − c(i)) ∗ rand()+c(i);
    end
end
M_OA = FitnessFunction(M_Ant);
M_OAL = FitnessFunction(M_Antlion);
M_Antlion = sort(M_Antlion);
M_OAL = sort(M_OAL);
Optimum = M_Antlion(1, :);
```

where $c(i)$ and $d(i)$ are the lower and upper bounds of the i-th parameter, respectively.

A.3. Matlab codes for the function B

```
Update ⃗c and ⃗d using Eqs. (2.10) and (2.11)
for i = 1: n
    Antlion = RouletteWheelSelection (M_Antlion, 1/M_OAL)
    Update c and d using Eqs. (2.10) and (2.11)
    c = Antlion + c
    d = Antlion + d
    c_elite = Elite + c
    d_elite = Elite + d
    for j = 1 : d
        R_A = X(t) in Eq. (2.1)
```

```
        R_A = normalize(R_A, c, d) in Eq. (2.7)
        R_E = X(t) in Eq. (2.1)
        R_E = normalize(R_E, c_elite ,d_elite) in Eq. (2.7)
        M_Ant (i,j) = (R_A + R_E)/2
    end
end
M_OA = FitnessFunction(M_Ant)
M_combined = concanenation of M_Ant and M_Antlion
M_combined = sort(M_combined)
M_Antlion = the first n rows of M_combined
If fintessfunction(M_Antlion(1,:)) < f(Elite)
    Elite = M_Antlion(1, :)
End
```

A.4. Appendix

See Table 19.

Appendix B. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.advengsoft.2015.01.010. Note that in the videos, the red triangles represent antlions and the blue circles show ants.

References

[1] Blum C, Puchinger J, Raidl GR, Roli A. Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput 2011;11:4135–51.
[2] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Inform Sci 2013;237:82–117.
[3] Gogna A, Tayal A. Metaheuristics: review and application. J Exp Theor Artif Intell 2013;25:503–26.
[4] Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ. A survey on metaheuristics for stochastic combinatorial optimization. Nat Comput: Int J 2009;8:239–87.
[5] Cornuéjols G. Valid inequalities for mixed integer linear programs. Math Program 2008;112:3–44.
[6] Avriel M. Nonlinear programming: analysis and methods. Courier Dover Publications; 2003.
[7] Land AH, Doig AG. An automatic method for solving discrete programming problems. In: 50 Years of integer programming 1958–2008. Springer; 2010. p. 105–32.
[8] Simpson AR, Dandy GC, Murphy LJ. Genetic algorithms compared to other techniques for pipe optimization. J Water Resour Plann Manage 1994;120:423–43.
[9] Spall JC. Introduction to stochastic search and optimization: estimation, simulation, and control, vol. 65. John Wiley & Sons; 2005.
[10] Back T. Evolutionary algorithms in theory and practice. Oxford Univ. Press; 1996.
[11] Hoos HH, Stützle T. Stochastic local search: foundations & applications. Elsevier; 2004.
[12] Kirkpatrick S. Optimization by simulated annealing: quantitative studies. J Stat Phys 1984;34:975–86.
[13] Talbi E-G. Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons; 2009.
[14] Holland JH. Genetic algorithms. Sci Am 1992;267:66–72.
[15] Holland JH, Reitman JS. Cognitive systems based on adaptive algorithms. ACM SIGART Bull 1977. p. 49–49.
[16] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science; 1995. p. 39–43.
[17] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life; 1991. p. 134–42.
[18] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341–59.
[19] Fogel LJ, Owens AJ, Walsh MJ. Artificial intelligence through simulated evolution; 1966.
[20] Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans Evol Comput 1999;3:82–102.
[21] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1:67–82.
[22] Mirjalili Seyedali, Mirjalili Seyed Mohammad, Lewis Andrew. Grey wolf optimizer. Adv Eng Software 2014;69:46–61.
[23] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71.

[24] Yang X-S. Firefly algorithm, Levy flights and global optimization. In: Research and development in intelligent systems XXVI. Springer; 2010. p. 209–18.
[25] Yang X-S. Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2010;2:78–84.
[26] Yang X-S, Deb S. Cuckoo search via Lévy flights. In: World congress on nature & biologically inspired computing, 2009. NaBIC 2009; 2009. p. 210–4.
[27] Yang X-S, Deb S. Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 2010;1:330–43.
[28] Rajabioun R. Cuckoo optimization algorithm. Appl Soft Comput 2011;11:5508–18.
[29] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm. Inform Sci 2009;179:2232–48.
[30] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mech 2010;213:267–89.
[31] Kaveh A, Talatahari S. Optimal design of skeletal structures via the charged system search algorithm. Struct Multidiscip Optim 2010;41:893–911.
[32] Kaveh A, Talatahari S. Charged system search for optimal design of frame structures. Appl Soft Comput 2012;12:382–93.
[33] Kaveh A, Talatahari S. Geometry and topology optimization of geodesic domes using charged system search. Struct Multidiscip Optim 2011;43:215–29.
[34] Kaveh A, Share MAM, Moslehi M. Magnetic charged system search: a new meta-heuristic algorithm for optimization. Acta Mech 2013;224:85–107.
[35] Kaveh A. Magnetic charged system search. In: Advances in metaheuristic algorithms for optimal design of structures. Springer; 2014. p. 87–134.
[36] Kaveh A, Ghazaan MI, Bakhshpoori T. An improved ray optimization algorithm for design of truss structures. Civ Eng 2013;57:97–112.
[37] Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization. Comput Struct 2012;112:283–94.
[38] Kaveh A, Khayatazad M. Ray optimization for size and shape optimization of truss structures. Comput Struct 2013;117:82–94.
[39] Kaveh A, Ghazaan MI. Enhanced colliding bodies optimization for design problems with continuous and discrete variables. Adv Eng Softw 2014; 77:66–75.
[40] Kaveh A, Ghazaan MI. Enhanced colliding bodies algorithm for truss optimization with frequency constraints. J Comput Civ Eng 2014.
[41] Kaveh A, Mahdavi V. Colliding bodies optimization method for optimum discrete design of truss structures. Comput Struct 2014;139:43–53.
[42] Kaveh A, Mahdavi V. Colliding bodies optimization method for optimum design of truss structures with continuous variables. Adv Eng Softw 2014; 70:1–12.
[43] Kaveh A, Mahdavi V. Colliding-bodies optimization for truss optimization with multiple frequency constraints. J Comput Civ Eng 2014.
[44] Kaveh A, Mahdavi V. Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 2014;139:18–27.
[45] Kaveh A, Bakhshpoori T, Afshari E. An efficient hybrid particle swarm and swallow swarm optimization algorithm. Comput Struct 2014;143:40–59.
[46] Kaveh A, Bijari S. Optimum cost design of reinforced concrete one-way ribbed slabs using CBO, PSO and democratic PSO algorithms. Asian J Civ Eng (BHRC) 2014;15:788–802.
[47] Kaveh A, Zolghadr A. Democratic PSO for truss layout and size optimization with frequency constraints. Comput Struct 2014;130:10–21.
[48] Kaveh A, Farhoudi N. A new optimization method: Dolphin echolocation. Adv Eng Softw 2013;59:53–70.
[49] Kaveh A. Dolphin echolocation optimization. In: Advances in metaheuristic algorithms for optimal design of structures. Springer; 2014. p. 157–93.
[50] Kaveh A, Sheikholeslami R, Talatahari S, Keshvari-Ilkhichi M. Chaotic swarming of particles: a new method for size optimization of truss structures. Adv Eng Softw 2014;67:136–47.
[51] Scharf I, Subach A, Ovadia O. Foraging behaviour and habitat selection in pit-building antlion larvae in constant light or dark conditions. Anim Behav 2008;76:2049–57.
[52] Griffiths D. Pit construction by ant-lion larvae: a cost-benefit analysis. J Anim Ecol 1986:39–57.
[53] Scharf I, Ovadia O. Factors influencing site abandonment and site selection in a sit-and-wait predator: a review of pit-building antlion larvae. J Insect Behav 2006;19:197–218.
[54] Grzimek B, Schlager N, Olendorf D, McDade MC. Grzimek's animal life encyclopedia. Michigan: Gale Farmington Hills; 2004.
[55] Goodenough J, McGuire B, Jakob E. Perspectives on animal behavior. John Wiley & Sons; 2009.
[56] Digalakis J, Margaritis K. On benchmarking functions for genetic algorithms. Int J Comput Math 2001;77:481–506.
[57] Molga M, Smutnicki C. Test functions for optimization needs. Test functions for optimization needs; 2005.
[58] Yang X-S. Test problems in optimization. arXiv preprint arXiv:1008.0549; 2010.
[59] Liang J, Suganthan P, Deb K. Novel composition test functions for numerical global optimization. In: Proceedings 2005 IEEE swarm intelligence symposium, 2005. SIS 2005; 2005. p. 68–75.
[60] Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y, Auger A, et al. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report, vol. 2005005; 2005.
[61] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings IEEE international conference on neural networks, 1995; 1995. p. 1942–8.
[62] John H. In: Holland, Adaptation natural and artificial systems. Cambridge, MA: MIT Press; 1992.
[63] Cuevas E, Echavarría A, Ramírez-Ortegón MA. An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. Appl Intell 2014;40:256–72.
[64] Cuevas E, Echavarría A, Zaldívar D, Pérez-Cisneros M. A novel evolutionary algorithm inspired by the states of matter for template matching. Expert Syst Appl 2013;40:6359–73.
[65] Yang X-S. A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer; 2010. p. 65–74.
[66] Yang X-S. Flower pollination algorithm for global optimization. In: Unconventional computation and natural computation. Springer; 2012. p. 240–9.
[67] Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 2011;1:3–18.
[68] Mirjalili S, Lewis A. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evolut Comput 2013;9:1–14.
[69] van den Bergh F, Engelbrecht A. A study of particle swarm optimization particle trajectories. Inform Sci 2006;176:937–71.
[70] Coello Coello CA. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 2002;191:1245–87.
[71] Chickermane H, Gea H. Structural optimization using a new local approximation method. Int J Numer Meth Eng 1996;39:829–46.
[72] Gandomi AH, Yang X-S, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 2013;29:17–35.
[73] Cheng M-Y, Prayogo D. Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 2014;139:98–112.
[74] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 2013;13:2592–612.
[75] Zhang M, Luo W, Wang X. Differential evolution with dynamic stochastic selection for constrained optimization. Inform Sci 2008;178:3043–74.
[76] Liu H, Cai Z, Wang Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 2010;10:629–40.
[77] Tsai J-F. Global optimization of nonlinear fractional programming problems in engineering design. Eng Optim 2005;37:399–409.
[78] Ray T, Saini P. Engineering design optimization using a swarm with an intelligent information sharing among individuals. Eng Optim 2001;33:735–48.
[79] Gandomi AH. Interior search algorithm (ISA): a novel approach for global optimization. ISA Trans 2014.
[80] Sandgren E. Nonlinear integer and discrete programming in mechanical design optimization. J Mech Des 1990;112:223–9.
[81] Wu S-J, Chow P-T. Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization. Eng Optim+ A35 1995;24:137–59.
[82] Sharma TK, Pant M, Singh V. Improved local search in artificial bee colony using golden section search. arXiv preprint arXiv:1210.6128; 2012.
[83] Deb K, Goyal M. A combined genetic adaptive search (GeneAS) for engineering design. Comput Sci Inform 1996;26:30–45.
[84] Kannan B, Kramer SN. An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J Mech Des 1994;116:405–11.
[85] Xie G. Optimal preliminary propeller design based on multi-objective optimization approach. Proc Eng 2011;16:278–83.
[86] Kim Y-C, Kim T-W, Pyo S, Suh J-C. Design of propeller geometry using streamline-adapted blade sections. J Mar Sci Technol 2009;14:161–70.
[87] Epps B, Chalfant J, Kimball R, Techet A, Flood K, Chryssostomidis C. OpenProp: an open-source parametric design and analysis tool for propellers. In: Proceedings of the 2009 grand challenges in modeling & simulation conference; 2009. p. 104–11.