Advances in Engineering Software 95 (2016) 51–67

Contents lists available at ScienceDirect

Advances in Engineering Software

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

The Whale Optimization Algorithm

Seyedali Mirjalili[a,b,*], Andrew Lewis[a]

[a] School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia
[b] Griffith College, Mt Gravatt, Brisbane, QLD 4122, Australia

ARTICLE INFO

Article history:
Received 7 August 2015
Revised 8 January 2016
Accepted 15 January 2016
Available online 26 February 2016

Keywords:
Optimization
Benchmark
Constrained optimization
Particle swarm optimization
Algorithm
Heuristic algorithm
Genetic algorithm
Structural optimization

ABSTRACT

This paper proposes a novel nature-inspired meta-heuristic optimization algorithm, called Whale Optimization Algorithm (WOA), which mimics the social behavior of humpback whales. The algorithm is inspired by the bubble-net hunting strategy. WOA is tested with 29 mathematical optimization problems and 6 structural design problems. Optimization results prove that the WOA algorithm is very competitive compared to the state-of-art meta-heuristic algorithms as well as conventional methods. The source codes of the WOA algorithm are publicly available at http://www.alimirjalili.com/WOA.html

1. Introduction

Meta-heuristic optimization algorithms are becoming more and more popular in engineering applications because they: (i) rely on rather simple concepts and are easy to implement; (ii) do not require gradient information; (iii) can bypass local optima; (iv) can be utilized in a wide range of problems covering different disciplines.

Nature-inspired meta-heuristic algorithms solve optimization problems by mimicking biological or physical phenomena. They can be grouped in three main categories (see Fig. 1): evolution-based, physics-based, and swarm-based methods. Evolution-based methods are inspired by the laws of natural evolution. The search process starts with a randomly generated population which is evolved over subsequent generations. The strength point of these methods is that the best individuals are always combined together to form the next generation of individuals. This allows the population to be optimized over the course of generations. The most popular evolution-inspired technique is Genetic Algorithms (GA) [1] that simulates the Darwinian evolution. Other popular algorithms are Evolution Strategy (ES) [110], Probability-Based Incremental Learning (PBIL) [111], Genetic Programming (GP) [2], and Biogeography-Based Optimizer (BBO) [3].

Physics-based methods imitate the physical rules in the universe. The most popular algorithms are Simulated Annealing (SA) [4,5], Gravitational Local Search (GLSA) [6], Big-Bang Big-Crunch (BBBC) [7], Gravitational Search Algorithm (GSA) [8], Charged System Search (CSS) [9], Central Force Optimization (CFO) [10], Artificial Chemical Reaction Optimization Algorithm (ACROA) [11], Black Hole (BH) [12] algorithm, Ray Optimization (RO) [13] algorithm, Small-World Optimization Algorithm (SWOA) [14], Galaxy-based Search Algorithm (GbSA) [15], and Curved Space Optimization (CSO) [16].

The third group of nature-inspired methods includes swarm-based techniques that mimic the social behavior of groups of animals. The most popular algorithm is Particle Swarm Optimization, originally developed by Kennedy and Eberhart [17]. PSO is inspired by the social behavior of bird flocking. It uses a number of particles (candidate solutions) which fly around in the search space to find the best solution (*i.e.* the optimal position). Meanwhile, they all trace the best location (best solution) in their paths. In other words, particles consider their own best solutions as well as the best solution the swarm has obtained so far. Another popular swarm-based algorithm is Ant Colony Optimization, first proposed by Dorigo et al. [18]. This algorithm is inspired by the social behavior of ants in an ant colony. In fact, the social intelligence of ants in finding the closest path from the nest and a source of food is the main inspiration of this algorithm. A pheromone matrix is evolved over the course of iteration by the candidate solutions.

Other swarm-based techniques are listed in Table 1. This class of meta-heuristic methods started to be attractive since PSO was

* Corresponding author. Tel.: +61 434555738.
E-mail address: seyedali.mirjalili@griffithuni.edu.au, ali.mirjalili@gmail.com (S. Mirjalili).

http://dx.doi.org/10.1016/j.advengsoft.2016.01.008
0965-9978/

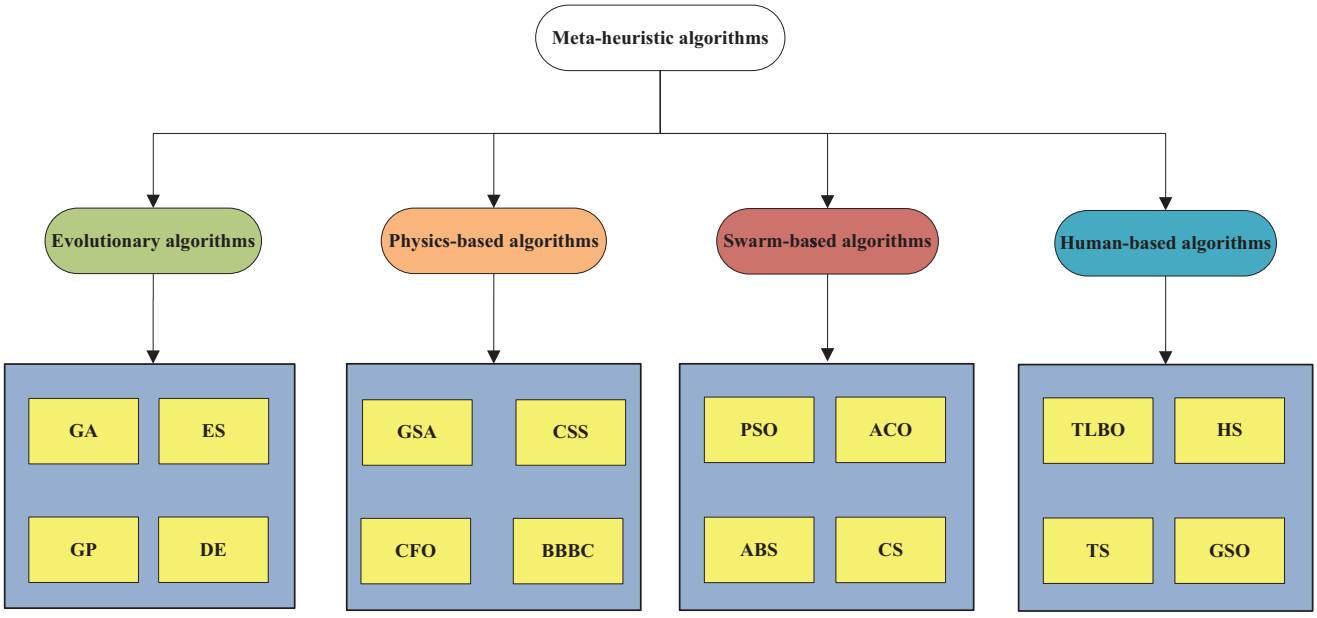

Fig. 1. Classification of meta-heuristic algorithms.

Table 1
Swarm-based optimization algorithms developed in literature.

Algorithm	Inspiration	Year of proposal
PSO [17]	Bird flock	1995
Marriage in Honey Bees Optimization Algorithm (MBO) [19]	Honey Bees	2001
Artificial Fish-Swarm Algorithm (AFSA) [20]	Fish swarm	2003
Termite Algorithm [21]	Termite colony	2005
ACO [18]	Ant colony	2006
ABC [22]	Honey Bee	2006
Wasp Swarm Algorithm [23]	Parasitic wasp	2007
Monkey Search [24]	Monkey	2007
Wolf pack search algorithm [25]	Wolf herd	2007
Bee Collecting Pollen Algorithm (BCPA) [26]	Bees	2008
Cuckoo Search (CS) [27]	Cuckoo	2009
Dolphin Partner Optimization (DPO) [28]	Dolphin	2009
Bat-inspired Algorithm (BA) [29]	Bat herd	2010
Firefly Algorithm (FA) [30]	Firefly	2010
Hunting Search (HS) [31]	Group of animals	2010
Bird Mating Optimizer (BMO) [32]	Bird mating	2012
Krill Herd (KH) [33]	Krill herd	2012
Fruit fly Optimization Algorithm (FOA) [34]	Fruit fly	2012
Dolphin Echolocation (DE) [35]	Dolphin	2013

proven to be very competitive with evolution-based and physical-based algorithms. Generally speaking, swarm-based algorithms have some advantages over evolution-based algorithms. For example, swarm-based algorithms preserve search space information over subsequent iterations while evolution-based algorithms discard any information as soon as a new population is formed. They usually include less operators compared to evolutionary approaches (selection, crossover, mutation, elitism, *etc.*) and hence are easier to implement.

It is worth mentioning here that there are also other meta-heuristic methods inspired by human behaviors in the literature. Some of the most popular algorithms are Teaching Learning Based Optimization(TLBO) [36,37], Harmony Search (HS) [38], Tabu (Taboo) Search (TS) [39–41], Group Search Optimizer (GSO) [42,43], Imperialist Competitive Algorithm (ICA) [44], League Championship Algorithm (LCA) [45,46], Firework Algorithm [47], Colliding Bodies Optimization (CBO) [48,49], Interior Search Algorithm (ISA) [50], Mine Blast Algorithm (MBA) [51], Soccer League Competition (SLC) algorithm [52,53], Seeker Optimization Algorithm (SOA) [54], Social-Based Algorithm (SBA) [55], Exchange Market Algorithm (EMA) [56], and Group Counseling Optimization (GCO) algorithm [57,58].

Population-based meta-heuristic optimization algorithms share a common feature regardless of their nature. The search process is divided into two phases: exploration and exploitation [59–61]. The optimizer must include operators to globally explore the search space: in this phase, movements (*i.e.* perturbation of design variables) should be randomized as most as possible. The exploitation phase follows the exploration phase and can be defined as the process of investigating in detail the promising area(s) of the search space. Exploitation hence pertains to the local search capability in the promising regions of design space found in the exploration phase. Finding a proper balance between exploration and exploitation is the most challenging task in the development of any meta-heuristic algorithm due to the stochastic nature of the optimization process.

This study describes a new meta-heuristic optimization algorithm (namely, Whale Optimization Algorithm, WOA) mimicking the hunting behavior of humpback whales. To the knowledge of the present authors, there is no previous study on this subject in the optimization literature. The main difference between the current work and the recently published papers by the authors (particularly GWO [62]) is the simulated hunting behavior with random or the best search agent to chase the prey and the use of a spiral to simulate bubble-net attacking mechanism of humpback whales. The efficiency of the WOA algorithm developed in this research is evaluated by solving 29 mathematical optimization problems and six structural optimization problems. Optimization results demonstrate that WOA is very competitive compared to the state-of-the-art optimization methods.

The rest of the paper is structured as follows. Section 2 describes the Whale Optimization Algorithm developed in this study. Test problems and optimization results are presented and discussed in Sections 3 and 4, respectively, for mathematical functions and structural optimization problems. Section 5 summarizes the main findings of this study and suggests directions for future research.

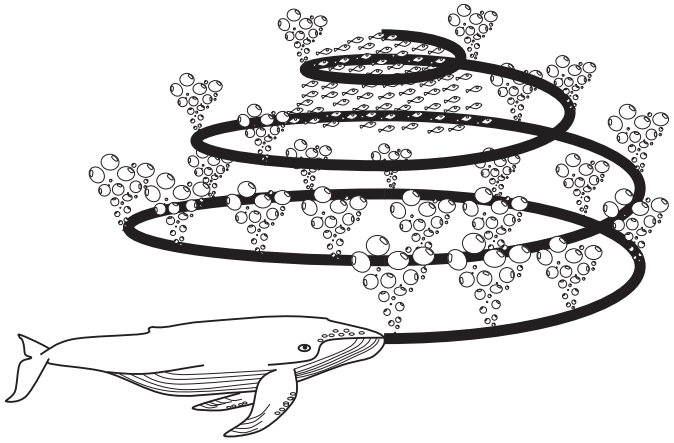

Fig. 2. Bubble-net feeding behavior of humpback whales.

2. Whale Optimization Algorithm (WOA)

In this section the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

2.1. Inspiration

Whales are fancy creatures. They are considered as the biggest mammals in the world. An adult whale can grow up to 30 m long and 180 t weight. There are 7 different main species of this giant mammal such killer, Minke, Sei, humpback, right, finback, and blue. Whales are mostly considered as predators. They never sleep because they have to breathe from the surface of oceans. In fact, half of the brain only sleeps. The interesting thing about the whales is that they are considered as highly intelligent animals with emotion.

According to Hof and Van Der Gucht [63], whales have common cells in certain areas of their brains similar to those of human called spindle cells. These cells are responsible for judgment, emotions, and social behaviors in humans. In other words the spindle cells make us distinct from other creatures. Whales have twice number of these cells than an adult human which is the main cause of their smartness. It has been proven that whale can think, learn, judge, communicate, and become even emotional as a human does, but obviously with a much lower level of smartness. It has been observed that whales (mostly killer whales) are able to develop their own dialect as well.

Another interesting point is the social behavior of whales. They live alone or in groups. However, they are mostly observed in groups. Some of their species (killer whales for instance) can live in a family over their entire life period. One of the biggest baleen whales is humpback whales (*Megaptera novaeangliae*). An adult humpback whale is almost as size of a school bus. Their favorite prey are krill and small fish herds. Fig. 2 shows this mammal.

The most interesting thing about the humpback whales is their special hunting method. This foraging behavior is called bubble-net feeding method [64]. Humpback whales prefer to hunt school of krill or small fishes close to the surface. It has been observed that this foraging is done by creating distinctive bubbles along a circle or '9'-shaped path as shown in Fig. 2. Before 2011, this behavior was only investigated based on the observation from surface. However, Goldbogen et al. [65] investigated this behavior utilizing tag sensors. They captured 300 tag-derived bubble-net feeding events of 9 individual humpback whales. They found two maneuvers associated with bubble and named them 'upward-spirals' and 'double-loops'. In the former maneuver, humpback whales dive around 12 m down and then start to create bubble in a spiral shape around the prey and swim up toward the surface. The later maneuver includes three different stages: coral loop, lobtail, and capture loop. Detailed information about these behaviors can be found in [65].

It is worth mentioning here that bubble-net feeding is a unique behavior that can only be observed in humpback whales. In this work the spiral bubble-net feeding maneuver is mathematically modeled in order to perform optimization.

2.2. Mathematical model and optimization algorithm

In this section the mathematical model of encircling prey, spiral bubble-net feeding maneuver, and search for prey is first provided. The WOA algorithm is then proposed.

2.2.1. Encircling prey

Humpback whales can recognize the location of prey and encircle them. Since the position of the optimal design in the search space is not known a priori, the WOA algorithm assumes that the current best candidate solution is the target prey or is close to the optimum. After the best search agent is defined, the other search agents will hence try to update their positions towards the best search agent. This behavior is represented by the following equations:

$$\vec{D} = |\vec{C}.\overrightarrow{X^*}(t) - \vec{X}(t)| \tag{2.1}$$

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} \tag{2.2}$$

where t indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, X^* is the position vector of the best solution obtained so far, $\vec{X}$ is the position vector, $|\ |$ is the absolute value, and $\cdot$ is an element-by-element multiplication. It is worth mentioning here that X^* should be updated in each iteration if there is a better solution.

The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{2.3}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{2.4}$$

where $\vec{a}$ is linearly decreased from 2 to 0 over the course of iterations (in both exploration and exploitation phases) and $\vec{r}$ is a random vector in [0,1].

Fig. 3(a) illustrates the rationale behind Eq. (2.2) for a 2D problem. The position (X,Y) of a search agent can be updated according to the position of the current best record (X^*,Y^*). Different places around the best agent can be achieved with respect to the current position by adjusting the value of $\vec{A}$ and $\vec{C}$ vectors. The possible updating position of a search agent in 3D space is also depicted in Fig. 3(b). It should be noted that by defining the random vector $(\vec{r})$ it is possible to reach any position in the search space located between the key-points shown in Fig. 3. Therefore, Eq. (2.2) allows any search agent to update its position in the neighborhood of the current best solution and simulates encircling the prey.

The same concept can be extended to a search space with n dimensions, and the search agents will move in hyper-cubes around the best solution obtained so far. As mentioned in the previous section, the humpback whales also attack the prey with the bubble-net strategy. This method is mathematically formulated as follows:

2.2.2. Bubble-net attacking method (exploitation phase)

In order to mathematically model the bubble-net behavior of humpback whales, two approaches are designed as follows:

1 *Shrinking encircling mechanism:* This behavior is achieved by decreasing the value of $\vec{a}$ in the Eq. (2.3). Note that the fluctuation range of $\vec{A}$ is also decreased by $\vec{a}$. In other words $\vec{A}$ is a

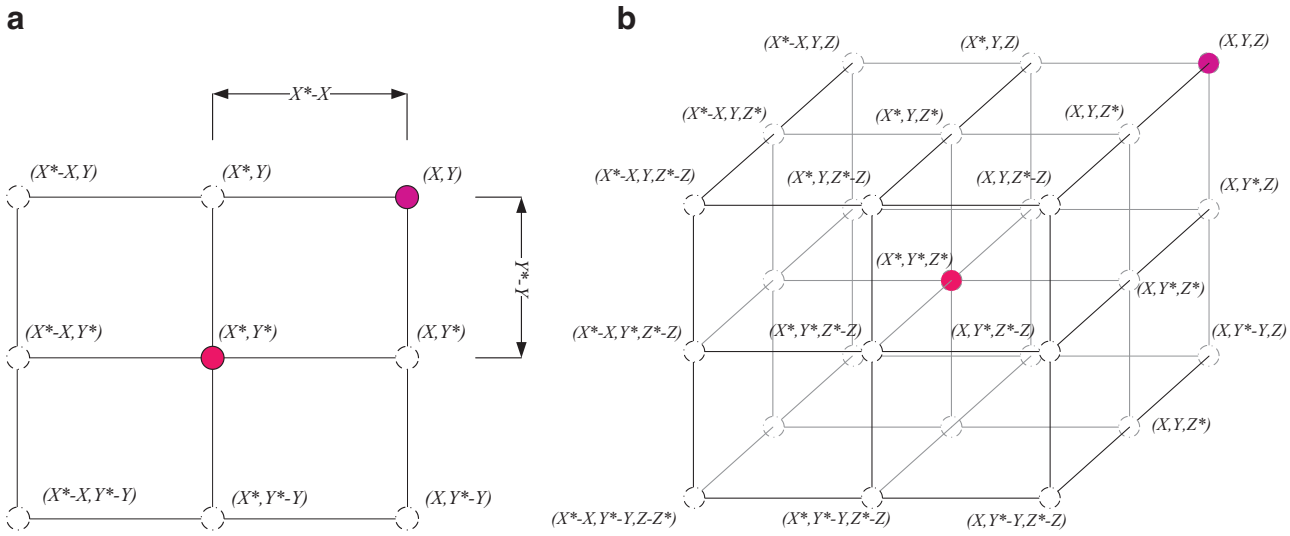

Fig. 3. 2D and 3D position vectors and their possible next locations (X^* is the best solution obtained so far).

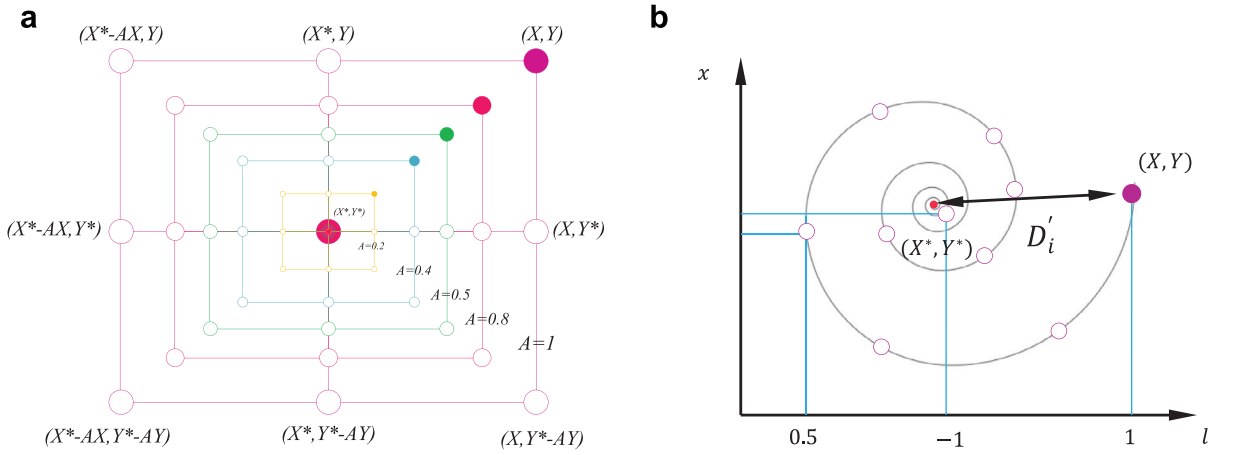

Fig. 4. Bubble-net search mechanism implemented in WOA (X^* is the best solution obtained so far): (a) shrinking encircling mechanism and (b) spiral updating position.

random value in the interval $[-a,a]$ where a is decreased from 2 to 0 over the course of iterations. Setting random values for $\vec{A}$ in $[-1,1]$, the new position of a search agent can be defined anywhere in between the original position of the agent and the position of the current best agent. Fig. 4(a) shows the possible positions from (X,Y) towards (X^*,Y^*) that can be achieved by $0 \leq A \leq 1$ in a 2D space.

2 *Spiral updating position:* As can be seen in Fig. 4(b), this approach first calculates the distance between the whale located at (X,Y) and prey located at (X^*,Y^*). A spiral equation is then created between the position of whale and prey to mimic the helix-shaped movement of humpback whales as follows:

$$\vec{X}(t+1) = \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) \tag{2.5}$$

where $\overrightarrow{D'} = |\overrightarrow{X^*}(t) - \vec{X}(t)|$ and indicates the distance of the ith whale to the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, l is a random number in $[-1,1]$, and . is an element-by-element multiplication.

Note that humpback whales swim around the prey within a shrinking circle and along a spiral-shaped path simultaneously. To model this simultaneous behavior, we assume that there is a probability of 50% to choose between either the shrinking encircling mechanism or the spiral model to update the position of whales during optimization. The mathematical model is as follows:

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} & if \ \ p < 0.5 \\ \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) & if \ \ p \geq 0.5 \end{cases} \tag{2.6}$$

where p is a random number in [0,1].

In addition to the bubble-net method, the humpback whales search for prey randomly. The mathematical model of the search is as follows.

2.2.3. Search for prey (exploration phase)

The same approach based on the variation of the $\vec{A}$ vector can be utilized to search for prey (exploration). In fact, humpback whales search randomly according to the position of each other. Therefore, we use $\vec{A}$ with the random values greater than 1 or less than −1 to force search agent to move far away from a reference whale. In contrast to the exploitation phase, we update the position of a search agent in the exploration phase according to a randomly chosen search agent instead of the best search agent found so far. This mechanism and $|\vec{A}| > 1$ emphasize exploration and allow the WOA algorithm to perform a global search. The mathematical model is as follows:

$$\vec{D} = |\vec{C} \cdot \overrightarrow{X_{rand}} - \vec{X}| \tag{2.7}$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}.\vec{D} \tag{2.8}$$

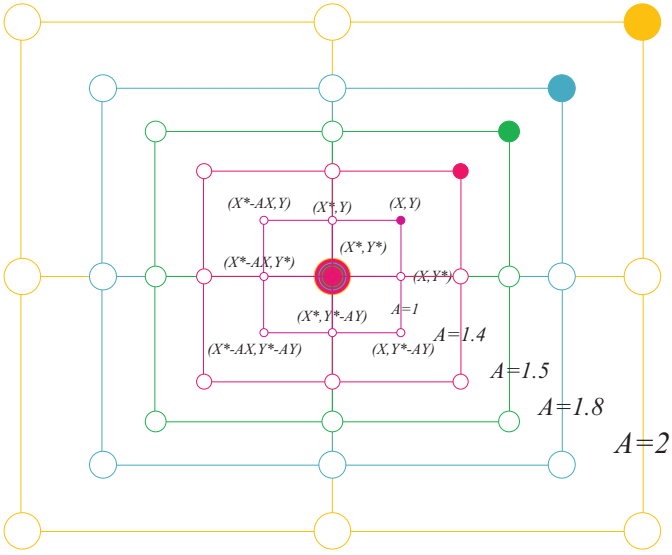

Fig. 5. Exploration mechanism implemented in WOA (X^* is a randomly chosen search agent).

```
Initialize the whales population Xi (i = 1, 2, ..., n)
Calculate the fitness of each search agent
X*=the best search agent
while (t < maximum number of iterations)
    for each search agent
    Update a, A, C, l, and p
        if1 (p<0.5)
            if2 (|A| < 1)
                Update the position of the current search agent by the Eq. (2.1)
            else if2 (|A| ≥ 1)
                Select a random search agent (X_rand)
                Update the position of the current search agent by the Eq. (2.8)
            end if2
        else if1 (p≥0.5)
                Update the position of the current search by the Eq. (2.5)
        end if1
    end for
    Check if any search agent goes beyond the search space and amend it
    Calculate the fitness of each search agent
    Update X* if there is a better solution
    t=t+1
end while
return X*
```

Fig. 6. Pseudo-code of the WOA algorithm.

where $\overrightarrow{X_{rand}}$ is a random position vector (a random whale) chosen from the current population.

Some of the possible positions around a particular solution with $\vec{A} > 1$ are depicted in Fig. 5.

The WOA algorithm starts with a set of random solutions. At each iteration, search agents update their positions with respect to either a randomly chosen search agent or the best solution obtained so far. The a parameter is decreased from 2 to 0 in order to provide exploration and exploitation, respectively. A random search agent is chosen when $|\vec{A}| > 1$, while the best solution is selected when $|\vec{A}| < 1$ for updating the position of the search agents. Depending on the value of p, WOA is able to switch between either a spiral or circular movement. Finally, the WOA algorithm is terminated by the satisfaction of a termination criterion.

The pseudo code of the WOA algorithm is presented in Fig. 6.

From theoretical stand point, WOA can be considered a global optimizer because it includes exploration/exploitation ability. Furthermore, the proposed hyper-cube mechanism defines a search space in the neighborhood of the best solution and allows other search agents to exploit the current best record inside that domain. Adaptive variation of the search vector A allows the WOA algorithm to smoothly transit between exploration and exploitation: by decreasing A, some iterations are devoted to exploration ($|A| \geq 1$) and the rest is dedicated to exploitation ($|A| < 1$). Remarkably, WOA includes only two main internal parameters to be adjusted (A and C).

Although mutation and other evolutionary operations might have been included in the WOA formulation to fully reproduce the behavior of humpback whales, we decided to minimize the amount of heuristics and the number of internal parameters thus implementing a very basic version of the WOA algorithm. However, hybridization with evolutionary search schemes may be the subject of future studies

Table 2
Description of unimodal benchmark functions.

Function	V_no	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10,10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	[−100,100]	0
$F_4(x) = \max_i\{\lvert x_i \rvert,\ 1 \leq i \leq n\}$	30	[−100,100]	0
$F_5(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[−30,30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100,100]	0
$F_7(x) = \sum_{i=1}^{n} ix_i^4 + random[0, 1)$	30	[−1.28,1.28]	0

3. Results and discussion

The numerical efficiency of the WOA algorithm developed in this study was tested by solving 29 mathematical optimization problems. The first 23 problems are classical benchmark functions utilized in the optimization literature [66–69]. WOA was compared with state-of-the-art swarm-based optimization algorithms. Tables 2–4 summarize the test problems reporting the cost function, range of variation of optimization variables and the optimal value f_{min} quoted in literature. The other 6 test problems considered in this study (see Table 5) regard composite benchmark functions considered in the CEC 2005 special session (see Ref. [70] for the detailed description of the composite functions). These benchmark functions are shifted, rotated, expanded, and combined variants of the most complicated mathematical optimization problems presented in literature [71]. Note that *V_no* indicates the number of design variables in Tables 2–5. For all the algorithms, a population size and maximum iteration equal to 30 and 500 have been utilized.

Generally speaking, benchmark functions can be divided into four groups: unimodal, multimodal, fixed-dimension multimodal, and composite functions. Fig. 7 shows the typical 2D plots of the cost function for some test cases considered in this study. It should be noted that the difference between fixed-dimensional multi-modal functions in Table 4 and multi-modal functions in Table 3 is the ability of defining the desired number of design variables. The mathematical formulations of fixed-dimensional test functions do not allow us to tune the number of design variables, but they provide different search space compared to multi-modal test functions in Table 3. It is also worth mentioning here that the composite test functions provide challenging test functions by shifting the global optimum to random positions before each run, locating the global optimum on the boundaries of the search spaces occasionally, and rotating the functions using $F(x) = f(R * x)$ formula where R is an orthogonal rotation matrices calculated using Salmon's method [72]. As Table 5 shows, composite test functions also combine different search spaces of n other primitive test functions by stretching and manipulating their coverage range. It should be noted that σ defines the coverage range of the primitive test functions and λ indicates the stretch/compression level of each of the primitive test functions.

Table 3
Description of multimodal benchmark functions.

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

Table 4
Description of fixed-dimenstion multimodal benchmark functions.

Function	V_no	Range	f_{min}
$F_{14}(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	2	[−65,65]	1
$F_{15}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	4	[−5,5]	0.00030
$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[−5,5]	−1.0316
$F_{17}(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	2	[−5,5]	0.398
$F_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[−2,2]	3
$F_{19}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$	3	[1,3]	−3.86
$F_{20}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	6	[0,1]	−3.32
$F_{21}(x)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	−10.1532
$F_{22}(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	−10.4028
$F_{23}(x)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	−10.5363

Table 5
Description of composite benchmark functions.

Function	V_no	Range	f_{min}
$F_{24}(CF1)$ $f_1, f_2, f_3, \ldots, f_{10}$ = Sphere Function $[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$ $[\lambda_1,\lambda_2,\lambda_3\ldots,\lambda_{10}]=[5/100,5/100,5/100,..,5/100]$	30	[−5,5]	0
$F_{25}(CF2)$ $f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank's Function* $[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$ $[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[5/100,5/100,5/100,..,5/100]$	30	[−5,5]	0
$F_{26}(CF3)$ $f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank's Function* $[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$ $[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[1,1,1,..,1]$	30	[−5,5]	0
$F_{27}(CF4)$ 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 $[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$ $[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[5/32,\ 5/32,\ 1,\ 1,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100,\ 5/100,5/100]$	30	[−5,5]	0
$F_{28}(CF5)$ 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 $[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[1,1,1,..,1]$ $[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[1/5,\ 1/5,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100,\ 5/32,\ 5/32,\ 5/100,\ 5/100]$	30	[−5,5]	0
$f_{29}(CF6$ 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 $[\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{10}]=[0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1]$ $[\lambda_1,\lambda_2,\lambda_3,\ldots,\lambda_{10}]=[0.1*1/5,\ 0.2*1/5,\ 0.3*5/0.5,\ 0.4*5/0.5,\ 0.5*5/100,$ $0.6*\ 5/100,\ 0.7*5/32, 0.8*\ 5/32,\ 0.9*5/100,\ 1*5/100]$	30	[−5,5]	0

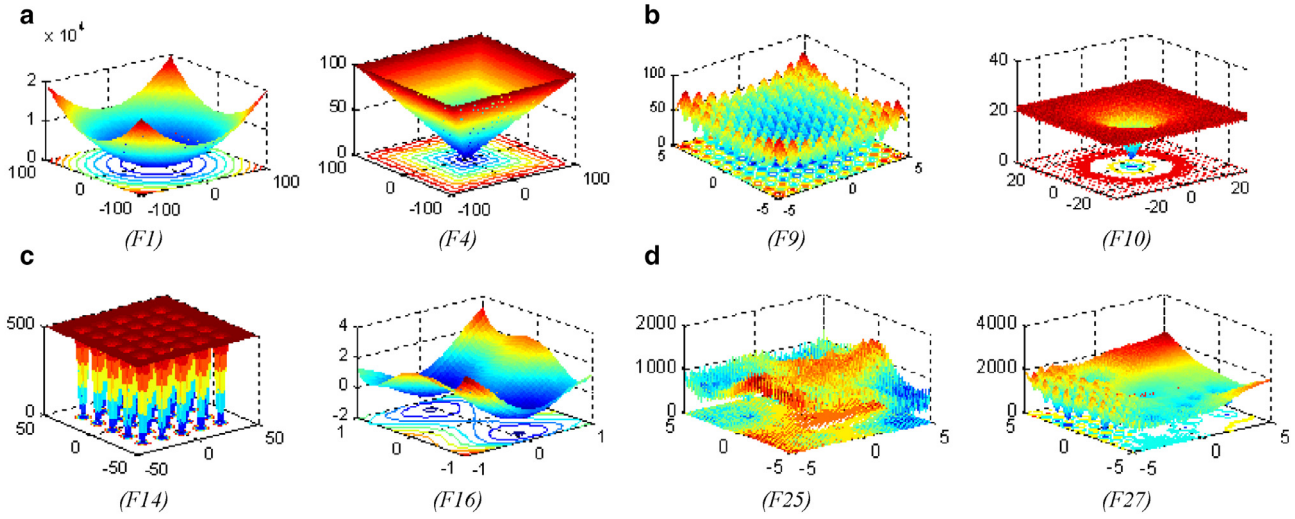

Fig. 7. Typical 2D representations of benchmark mathematical functions: (a) Unimodal functions; (b) Multimodal functions; (c) fixed-dimension multimodal functions; and (d) composite mathematical functions.

Table 6
Comparison of optimization results obtained for the unimodal, multimodal, and fixed-dimension multimodal benchmark functions.

F	WOA		PSO		GSA		DE		FEP	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F1	1.41E−30	4.91E−30	0.000136	0.000202	2.53E−16	9.67E−17	8.2E−14	5.9E−14	0.00057	0.00013
F2	1.06E−21	2.39E−21	0.042144	0.045421	0.055655	0.194074	1.5E−09	9.9E−10	0.0081	0.00077
F3	5.39E−07	2.93E−06	70.12562	22.11924	896.5347	318.9559	6.8E−11	7.4E−11	0.016	0.014
F4	0.072581	0.39747	1.086481	0.317039	7.35487	1.741452	0	0	0.3	0.5
F5	27.86558	0.763626	96.71832	60.11559	67.54309	62.22534	0	0	5.06	5.87
F6	3.116266	0.532429	0.000102	8.28E−05	2.5E−16	1.74E−16	0	0	0	0
F7	0.001425	0.001149	0.122854	0.044957	0.089441	0.04339	0.00463	0.0012	0.1415	0.3522
F8	−5080.76	695.7968	−4841.29	1152.814	−2821.07	493.0375	−11080.1	574.7	−12554.5	52.6
F9	0	0	46.70423	11.62938	25.96841	7.470068	69.2	38.8	0.046	0.012
F10	7.4043	9.897572	0.276015	0.50901	0.062087	0.23628	9.7E−08	4.2E−08	0.018	0.0021
F11	0.000289	0.001586	0.009215	0.007724	27.70154	5.040343	0	0	0.016	0.022
F12	0.339676	0.214864	0.006917	0.026301	1.799617	0.95114	7.9E−15	8E−15	9.2E−06	3.6E−06
F13	1.889015	0.266088	0.006675	0.008907	8.899084	7.126241	5.1E−14	4.8E−14	0.00016	0.000073
F14	2.111973	2.498594	3.627168	2.560828	5.859838	3.831299	0.998004	3.3E−16	1.22	0.56
F15	0.000572	0.000324	0.000577	0.000222	0.003673	0.001647	4.5E−14	0.00033	0.0005	0.00032
F16	−1.03163	4.2E−07	−1.03163	6.25E−16	−1.03163	4.88E−16	−1.03163	3.1E−13	−1.03	4.9E−07
F17	0.397914	2.7E−05	0.397887	0	0.397887	0	0.397887	9.9E−09	0.398	1.5E−07
F18	3	4.22E−15	3	1.33E−15	3	4.17E−15	3	2E−15	3.02	0.11
F19	−3.85616	0.002706	−3.86278	2.58E−15	−3.86278	2.29E−15	N/A	N/A	−3.86	0.000014
F20	−2.98105	0.376653	−3.26634	0.060516	−3.31778	0.023081	N/A	N/A	−3.27	0.059
F21	−7.04918	3.629551	−6.8651	3.019644	−5.95512	3.737079	−10.1532	0.0000025	−5.52	1.59
F22	−8.18178	3.829202	−8.45653	3.087094	−9.68447	2.014088	−10.4029	3.9E−07	−5.53	2.12
F23	−9.34238	2.414737	−9.95291	1.782786	−10.5364	2.6E−15	−10.5364	1.9E−07	−6.57	3.14

Table 7
Comparison of optimization results obtained for the composite benchmark functions.

F	WOA		PSO		GSA		DE		CMA-ES	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F24	0.568846	0.505946	100	81.65	6.63E−17	2.78E−17	6.75E−2	1.11E−1	100	188.56
F25	75.30874	43.07855	155.91	13.176	200.6202	67.72087	28.759	8.6277	161.99	151
F26	55.65147	21.87944	172.03	32.769	180	91.89366	144.41	19.401	214.06	74.181
F27	53.83778	21.621	314.3	20.066	170	82.32726	324.86	14.784	616.4	671.92
F28	77.8064	52.02346	83.45	101.11	200	47.14045	10.789	2.604	358.3	168.26
F29	57.88445	34.44601	861.42	125.81	142.0906	88.87141	490.94	39.461	900.26	8.32E−02

For each benchmark function, the WOA algorithm was run 30 times starting from different populations randomly generated. Statistical results (*i.e.* average cost function and corresponding standard deviation) are reported in Tables 6 and 7. WOA was compared with PSO [17], GSA [8], DE [73], Fast Evolutionary Programing (FEP) [66], and Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [74]. Note that most of the results of the comparative algorithms are taken from [62].

3.1. Evaluation of exploitation capability (functions F1–F7)

Functions F1–F7 are unimodal since they have only one global optimum. These functions allow to evaluate the exploitation capability of the investigated meta-heuristic algorithms. It can be seen from Table 6 that WOA is very competitive with other meta-heuristic algorithms. In particular, it was the most efficient optimizer for functions F1 and F2 and at least the second best

optimizer in most test problems. The present algorithm can hence provide very good exploitation.

3.2. Evaluation of exploration capability (functions F8–F23)

Unlike unimodal functions, multimodal functions include many local optima whose number increases exponentially with the problem size (number of design variables). Therefore, this kind of test problems turns very useful if the purpose is to evaluate the exploration capability of an optimization algorithm. The results reported in Table 6 for functions F8–F23 (*i.e.* multimodal and fixed-dimension multimodal functions) indicate that WOA has also a very good exploration capability. In fact, the present algorithm always is the most efficient or the second best algorithm in the majority of test problems. This is due to integrated mechanisms of exploration in the WOA algorithm that leads this algorithm towards the global optimum.

3.3. Ability to escape from local minima (functions F24–F29)

Optimization of composite mathematical functions is a very challenging task because only a proper balance between exploration and exploitation allows local optima to be avoided. Optimization results reported in Table 7 show that the WOA algorithm was the best optimizer in three test problems and was very competitive in the other cases. This proves that WOA can well balance exploration and exploitation phases. Such ability derives from the adaptive strategy utilized to update the A vector: some iterations are devoted to exploration ($|A| \geq 1$) while the rest to exploitation ($|A| < 1$).

3.4. Analysis of convergence behavior

It was observed that WOA's search agents tend to extensively search promising regions of design space and exploit the best one. Search agents change abruptly in the early stages of the optimization process and then gradually converge. According to Berg et al. [75], such a behavior can guarantee that a population-based algorithm eventually convergences to a point in a search space. Convergence curves of WOA, PSO and GSA are compared in Fig. 8 for some of the problems. It can be seen that WOA is enough competitive with other state-of-the-art meta-heuristic algorithms.

The convergence curves of the WOA, PSO, and GSA are provided in Fig. 8 to see the convergence rate of the algorithms. Please note that average best-so-far indicates the average of the best solution obtained so far in each iteration over 30 runs. As may be seen in this figure, the WOA algorithm shows three different convergence behaviors when optimizing the test functions. Firstly, the convergence of the WOA algorithm tends to be accelerated as iteration increases. This is due to the adaptive mechanism proposed for WOA that assists it to look for the promising regions of the search space in the initial steps of iteration and more rapidly converge towards the optimum after passing almost half of the iterations. This behavior is evident in F1, F3, F4, and F9. The second behavior is the convergence towards the optimum only in final iterations as may be observed in F8 and F21. This is probably due to the failure of WOA in finding a good solution for exploitation in the initial steps of iteration when avoiding local optima, so this algorithm keep searching the search space to find good solutions

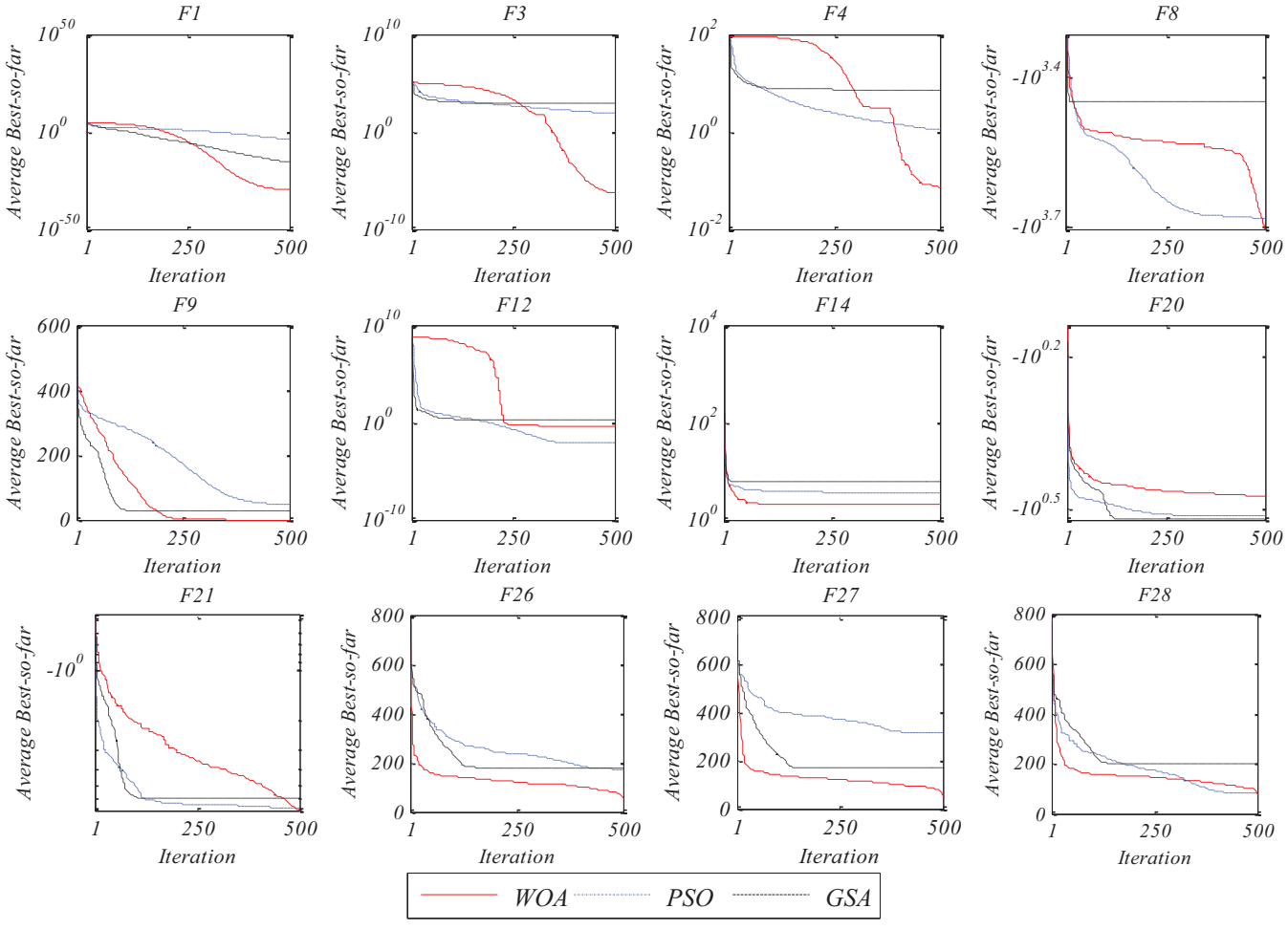

Fig. 8. Comparison of convergence curves of WOA and literature algorithms obtained in some of the benchmark problems.

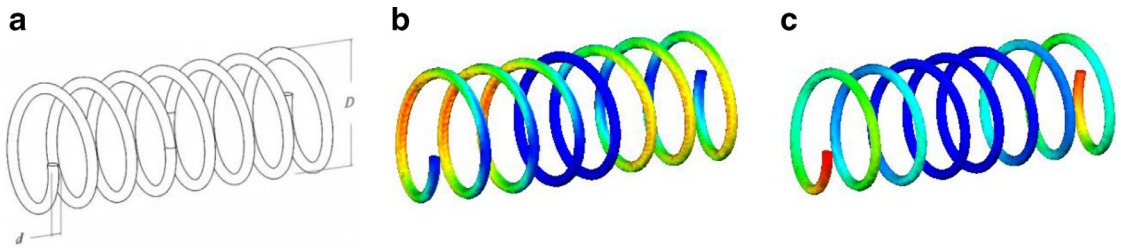

Fig. 9. (a) Schematic of the spring; (b) stress distribution evaluated at the optimum design; and (c) displacement distribution evaluated at the optimum design.

to converge toward them. The last behavior is the rapid convergence from the initial steps of iterations as can be seen in F14, F26, F27, and F28. Since F26, F27, and F28 are the most challenging test beds of this section (composite test functions), these results show that the WOA algorithm benefits from a good balance of exploration and exploitation that assists this algorithm to find the global optima. Overall, it seems the success rate of the WOA algorithm is high in solving challenging problems.

As a summary, the results of this section revealed different characteristics of the proposed WOA algorithm. High exploration ability of WOA is due to the position updating mechanism of whales using Eq. (2.8). This equation requires whales to move randomly around each other during the initial steps of the iterations. In the rest of iterations, however, high exploitation and convergence are emphasized which originate from Eq. (2.6). This equation allows the whales to rapidly re-position themselves around or move in spiral-shaped path towards the best solution obtained so far. Since these two phases are done separately and in almost half of iterations each, the WOA shows high local optima avoidance and convergence speed simultaneously during the course of iterations. However, PSO and GSA do not have operators to devote specific iterations to exploration or exploitation. In other words, PSO and GSA (and any other similar algorithms) utilize one formula to update the position of search agents, which increase the likeliness of stagnation in local optima. In the following sections the performance of WOA is verified on more challenging real engineering problems.

4. WOA for classical engineering problems

In this section, WOA was tested also with six constrained engineering design problems: a tension/compression spring, a welded beam, a pressure vessel, a 15-bar truss, a 25-bar truss, and a 52-bar truss.

Since the problems of this section have different constraints, we need to employ a constraint handling method. There are different types of penalty functions in the literature [76]: static, dynamic, annealing, adaptive, co-evolutionary, and death penalty. The last penalty function, death penalty, is the simplest method, which assigns a big objective value (in case of minimization). This process automatically causes discarding the infeasible solutions by the heuristic algorithms during optimization. The advantages of this method are simplicity and low computational cost. However, this method does not utilize the information of infeasible solutions that might be helpful when solving problems with dominated infeasible regions. For the sake of simplicity, we equip the WOA algorithm with a death penalty function in this section to handle constraints.

4.1. Tension/compression spring design

The objective of this test problem is to minimize the weight of the tension/compression spring shown in Fig. 9 [77–79]. Optimum design must satisfy constraints on shear stress, surge frequency and deflection. There are three design variables: wire diameter (d), mean coil diameter (D), and number of active coils (N). The optimization problem is formulated as follows:

$$
\begin{aligned}
&\text{Consider} && \vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N],\\
&\text{Minimize} && f(\vec{x}) = (x_3+2)x_2x_1^2\\
&\text{Subject to} && g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0,\\
& && g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} \le 0,\\
& && g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0,\\
& && g_4(\vec{x}) = \frac{x_1+x_2}{1.5} - 1 \le 0,\\
&\text{Variable range} && 0.05 \le x_1 \le 2.00,\\
& && 0.25 \le x_2 \le 1.30,\\
& && 2.00 \le x_3 \le 15.0
\end{aligned}
\qquad (5.1)
$$

Table 8
Comparison of WOA optimization results with literature for the tension/compression spring design problem.

Algorithm	Optimum variables			Optimum
	d	D	N	weight
WOA	0.051207	0.345215	12.004032	0.0126763
GSA	0.050276	0.323680	13.525410	0.0127022
PSO (Ha and Wang)	0.051728	0.357644	11.244543	0.0126747
ES (Coello and Montes)	0.051989	0.363965	10.890522	0.0126810
GA (Coello)	0.051480	0.351661	11.632201	0.0127048
RO (Kaveh and Khayatazad)	0.051370	0.349096	11.76279	0.0126788
Improved HS (Mahdavi et al.)	0.051154	0.349871	12.076432	0.0126706
DE (Huang et al.)	0.051609	0.354714	11.410831	0.0126702
Mathematical optimization (Belegundu)	0.053396	0.399180	9.1854000	0.0127303
Constraint correction (Arora)	0.050000	0.315900	14.250000	0.0128334

Table 9
Comparison of WOA statistical results with literature for the tension/compression design problem.

Algorithm	Average	Standard deviation	Function evaluation
WOA	0.0127	0.0003	4410
PSO	0.0139	0.0033	5460
GSA	0.0136	0.0026	4980

This test case was solved using either mathematical techniques (for example, constraints correction at constant cost [77] and penalty functions [78]) or meta-heuristic techniques such as PSO [80], Evolution Strategy (ES) [81], GA [82], improved Harmony Search (HS) [83], and Differential Evolution (DE) [84], and Ray Optimization (RO) algorithm [13].

Optimization results of WOA are compared with literature in Table 8. A different penalty function constraint handling strategy was utilized in order to perform a fair comparison with literature [85]. It can be seen that WOA outperforms all other algorithms except HS and DE.

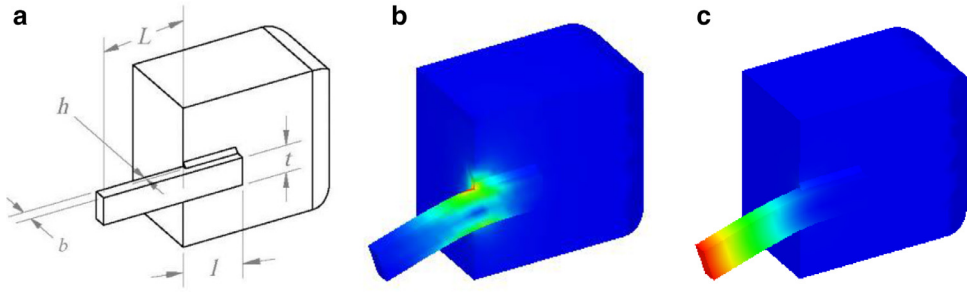

Fig. 10. Welded beam design problem: (a) Schematic of the weld; (b) Stress distribution evaluated at the optimum design; (c) Displacement distribution at the optimum design.

Table 9 also includes the average, standard deviation, and number of analysis by three of the algorithms over 30 runs. Note that we have utilized 10 search agents and a maximum number of 500 iterations to solve this problem. This table shows that the WOA algorithm outperforms PSO and GSA in average and requires less number of analyses (function evaluation).

4.2. Welded beam design

The objective of this test problem is to minimize the fabrication cost of the welded beam shown in Fig. 10 [82]. Optimization constraints are on shear stress (τ) and bending stress in the beam (θ), buckling load (P_c), end deflection of the beam (δ). There are four optimization variables: thickness of weld (h), length of the clamped bar (l), height of the bar (t), and thickness of the bar (b). The mathematical formulation of the optimization problem is as follows:

$$
\begin{aligned}
\text{Consider}\quad & \vec{x} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b],\\
\text{Minimize}\quad & f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2),\\
\text{Subject to}\quad & g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \le 0,\\
& g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \le 0,\\
& g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \le 0,\\
& g_4(\vec{x}) = x_1 - x_4 \le 0,\\
& g_5(\vec{x}) = P - P_c(\vec{x}) \le 0,\\
& g_6(\vec{x}) = 0.125 - x_1 \le 0\\
& g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2)\\
& \qquad - 5.0 \le 0 \qquad (5.2)\\
\text{Variable range}\quad & 0.1 \le x_1 \le 2,\\
& 0.1 \le x_2 \le 10,\\
& 0.1 \le x_3 \le 10,\\
& 0.1 \le x_4 \le 2
\end{aligned}
$$

where

$$
\begin{aligned}
& \tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\\
& \tau' = \frac{P}{\sqrt{2}x_1x_2},\quad \tau'' = \frac{MR}{J},\quad M = P\left(L + \frac{x_2}{2}\right),\\
& R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},\\
& J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},\\
& \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2},\ \delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}\\
& P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\\
& P = 6000\,\text{lb},\ L = 14\,\text{in.},\ \delta_{max} = 0.25\ \text{in.},\\
& E = 30 \times 1^6\,\text{psi},\ G = 12 \times 10^6\,\text{psi},\\
& \tau_{max} = 13,600\,\text{psi},\ \sigma_{max} = 30,000\,\text{psi},
\end{aligned}
$$

This optimization problem was solved by Coello [86] and Deb [87,88] with GA while Lee and Geem [89] utilized HS, Mahdavi et al. used an improved HS [83], RO was employed by Kaveh and Khayatzad [13], and Kaveh and Mahdavi solved this problem using CBO [49]. Richardson's random method, Simplex method, Davidon–Fletcher–Powell, Griffith and Stewart's successive linear approximation are the mathematical approaches adopted by Radgsdell and Philips [90]. Optimization results given in Table 10 indicate that WOA converged to third best design.

Table 11 contains the statistical results of some of the algorithms over 30 independent runs. We have utilized 20 search agents and a maximum number of 500 iterations to solve this problem. It may be observed in this table that WOA again shows

Table 10
Comparison of WOA optimization results with literature for the welded beam design problem.

Algorithm	Optimum variables				Optimum
	h	l	t	b	cost
WOA	0.205396	3.484293	9.037426	0.206276	1.730499
GSA	0.182129	3.856979	10.00000	0.202376	1.879952
CBO	0.205722	3.47041	9.037276	0.205735	1.724663
RO	0.203687	3.528467	9.004233	0.207241	1.735344
Improved HS	0.20573	3.47049	9.03662	0.2057	1.7248
GA Coello)	N/A	N/A	N/A	N/A	1.8245
GA (Deb)	N/A	N/A	N/A	N/A	2.3800
GA (Deb)	0.2489	6.1730	8.1789	0.2533	2.4331
HS (Lee and Geem)	0.2442	6.2231	8.2915	0.2443	2.3807
Random	0.4575	4.7313	5.0853	0.6600	4.1185
Simplex	0.2792	5.6256	7.7512	0.2796	2.5307
David	0.2434	6.2552	8.2915	0.2444	2.3841
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815

Table 11
Comparison of WOA statistical results with literature for the welded beam design problem.

Algorithm	Average	Standard deviation	Function evaluation
WOA	1.7320	0.0226	9900
PSO	1.7422	0.01275	13770
GSA	3.5761	1.2874	10750

better performance in average. In addition, this algorithm needs the least number of analyses to find the best optimal design.

4.3. Pressure vessel design

In this problem the goal is to minimize the total cost (material, forming and welding) of the cylindrical pressure vessel shown in Fig. 11. Both ends of the vessel are capped while the head has a hemi-spherical shape. There are four optimization variables: the thickness of the shell (T_s), the thickness of the head (T_h), the inner radius (R), the length of the cylindrical section without considering the head (L). The problem includes four optimization constraints and is formulated as follows:

$$
\begin{aligned}
\text{Consider}\quad & \vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L],\\
\text{Minimize}\quad & f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2\\
& \quad + 3.1661x_1^2x_4 + 19.84x_1^2x_3,\\
\text{Subject to}\quad & g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0,\\
& g_2(\vec{x}) = -x_3 + 0.00954x_3 \le 0,\\
& g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0,\\
& g_4(\vec{x}) = x_4 - 240 \le 0, \qquad (5.3)\\
\text{Variable range}\quad & 0 \le x_1 \le 99,\\
& 0 \le x_2 \le 99,\\
& 10 \le x_3 \le 200,\\
& 10 \le x_4 \le 200,
\end{aligned}
$$

This test case was solved by many researchers with meta-heuristic methods (for example, PSO [80], GA [79,82,91], ES [81], DE [84], ACO [92], and improved HS [83]) or mathematical methods (for example, Lagrangian multiplier [93] and branch-and-bound [94]).

Optimization results are compared with literature in Table 12. WOA was the third most efficient optimizer after ACO and DE. However, we found that the ACO violated one of the constraints. Therefore, it can be stated that the WOA is able to find the second best feasible optimal design for the pressure vessel design problem.

The statistical results of some of the algorithms when solving the pressure design problem are presented in Table 13. We have used 20 search agents and a maximum number of 500 iterations to solve this problem. According to the results in this table, once more, the WOA outperforms the PSO and GSA algorithms. In addition Table 13 shows that the WOA finds the best optimal design with the least number of function evaluations.

Table 13
Comparison of WOA statistical results with literature for the pressure vessel design problem.

Algorithm	Average	Standard deviation	Function evaluation
WOA	6068.05	65.6519	6300
PSO	6531.10	154.3716	14790
GSA	8932.95	683.5475	7110

4.4. 15-bar truss design

This problem is a discrete problem, in which the objective is to minimize the weight of a 15-bar truss. The final optimal design for this problem should satisfy 46 constraints such as 15 tension, 15 compression, and 16 displacement constraints. There are also 8 nodes and 15 bars as shown in Fig. 12, so there is the total number of 15 variables. It also may be seen in this figure that three loads are applied to the nodes $P1$, $P2$, and $P3$. Other assumptions for this problem are as follows:

- $\rho = 7800\,\text{kg/m}^3$
- $E = 200\,\text{GPa}$

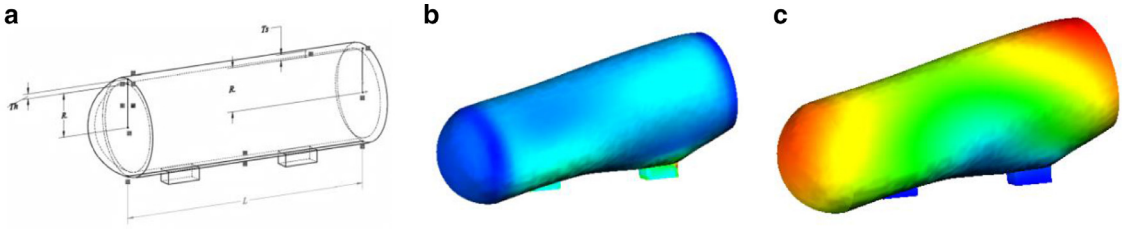

Fig. 11. Pressure vessel design problem: (a) schematic of the vessel; (b) stress distribution evaluated at the optimum design; and (c) displacement distribution evaluated at the optimum design.

Table 12
Comparison of WOA optimization results with literature for the pressure vessel design problem.

Algorithm	Optimum variables				Optimum cost
	T_s	T_h	R	L	
WOA	0.812500	0.437500	42.0982699	176.638998	6059.7410
Improved HS	1.125000	0.625000	58.29015	43.69268	7197.730
GSA	1.125000	0.625000	55.9886598	84.4542025	8538.8359
PSO (He and Wang)	0.812500	0.437500	42.091266	176.746500	6061.0777
GA (Coello)	0.812500	0.434500	40.323900	200.000000	6288.7445
GA (Coello and Montes)	0.812500	0.437500	42.097398	176.654050	6059.9463
GA (Deb and Gene)	0.937500	0.500000	48.329000	112.679000	6410.3811
ES (Montes and Coello)	0.812500	0.437500	42.098087	176.640518	6059.7456
DE (Huang et al.)	0.812500	0.437500	42.098411	176.637690	6059.7340
ACO (Kaveh and Talataheri)	0.812500	0.437500	42.103624	176.572656	6059.0888 (infeasible)
Lagrangian multiplier (Kannan)	1.125000	0.625000	58.291000	43.6900000	7198.0428
Branch-bound (Sandgren)	1.125000	0.625000	47.700000	117.701000	8129.1036

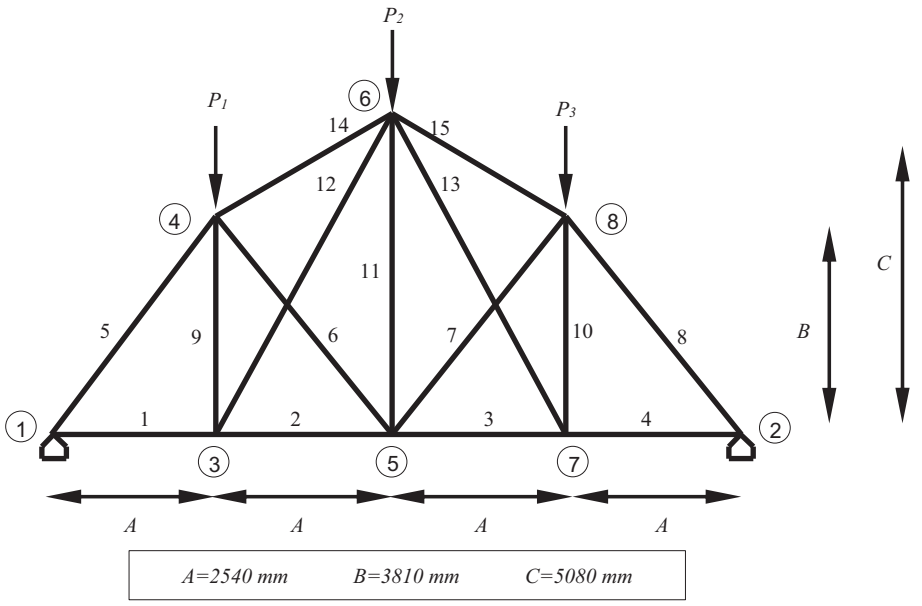

Fig. 12. Structure of a 15-bar truss.

Table 14
Comparison of WOA optimization results with literature for the 15-bar truss design problem.

Variables (mm^2)	HGA [97]	PSO [31]	PSOPC [31]	HPSO [31]	MBA [96]	SOS [98]	WOA
A1	308.6	185.9	113.2	113.2	113.2	113.2	113.2
A2	174.9	113.2	113.2	113.2	113.2	113.2	113.2
A3	338.2	143.2	113.2	113.2	113.2	113.2	113.2
A4	143.2	113.2	113.2	113.2	113.2	113.2	113.2
A5	736.7	736.7	736.7	736.7	736.7	736.7	736.7
A6	185.9	143.2	113.2	113.2	113.2	113.2	113.2
A7	265.9	113.2	113.2	113.2	113.2	113.2	113.2
A8	507.6	736.7	736.7	736.7	736.7	736.7	736.7
A9	143.2	113.2	113.2	113.2	113.2	113.2	113.2
A10	507.6	113.2	113.2	113.2	113.2	113.2	113.2
A11	279.1	113.2	113.2	113.2	113.2	113.2	113.2
A12	174.9	113.2	113.2	113.2	113.2	113.2	113.2
A13	297.1	113.2	185.9	113.2	113.2	113.2	113.2
A14	235.9	334.3	334.3	334.3	334.3	334.3	334.3
A15	265.9	334.3	334.3	334.3	334.3	334.3	334.3
Optimal weight (kg)	142.117	108.84	108.96	105.735	105.735	105.735	105.735
No. of analyses	N/A	~18,500	~16,000	7500	2000	5000	4000

- Stress limitation = 120 MPa
- Maximum stress = 115.37 MPa
- Displacement limitation = 10 mm
- Maximum displacement = 4.24 mm
- Design variabe set = $\left\{\begin{matrix} 113.2,\ 143.2,\ 145.9,\ 174.9,\ 185.9, \\ 235.9,\ 265.9,\ 297.1 \\ 308.6,\ 334.3,\ 338.2,\ 497.8,\ 507.6, \\ 736.7,\ 791.2,\ 1063.7 \end{matrix}\right\}$

This problem has been solved with three different set of loads (cases) in the literature as follows [95,96]:

- *Case* 1 : $P_1 = 35\,\text{kN},\ P_2 = 35\,\text{kN},\ P_3 = 35\,\text{kN}$
- *Case* 2 : $P_1 = 35\,\text{kN},\ P_2 = 0\,\text{kN},\ P_3 = 35\,\text{kN}$
- *Case* 3 : $P_1 = 35\,\text{kN},\ P_2 = 0\,\text{kN},\ P_3 = 0\,\text{kN}$

We solve these three cases using 30 search agents over 500 iterations and present the results in Table 14. Since this problem is a discrete problem, the search agents of WOA were simply rounded to the nearest integer number during optimization.

Table 14 shows that the WOA algorithm is able to find a similar structure compared to those of HPSO, SOS, and MBA. This shows that this algorithm is able to provide very competitive results in solving this problem as well. In addition, the number of function evaluations of WOA ranks second after the MBA algorithm.

4.5. 25-bar truss design

This problem is another popular truss design problem in the literature. As may be seen in Fig. 13, there are 10 nodes of which four are fixed. There are also 25 bars (cross-sectional areas members), which are classified in 8 groups as follows:

- *Group* 1 : A_1
- *Group* 2 : $A_2,\ A_3,\ A_4,\ A_5$
- *Group* 3 : $A_6,\ A_7,\ A_8,\ A_9$
- *Group* 4 : $A_{10},\ A_{11}$
- *Group* 5 : $A_{12},\ A_{13}$
- *Group* 6 : $A_{14},\ A_{15},\ A_{17}$
- *Group* 7 : $A_{18},\ A_{19},\ A_{20}, A_{21}$
- *Group* 8 : $A_{22},\ A_{23},\ A_{24},\ A_{25}$

Therefore, this problem has 8 parameters. Other assumptions for this problem are as follows:

- $\rho = 0.0272\,\text{N/cm}^3\ (0.1\,\text{lb/in.}^3)$

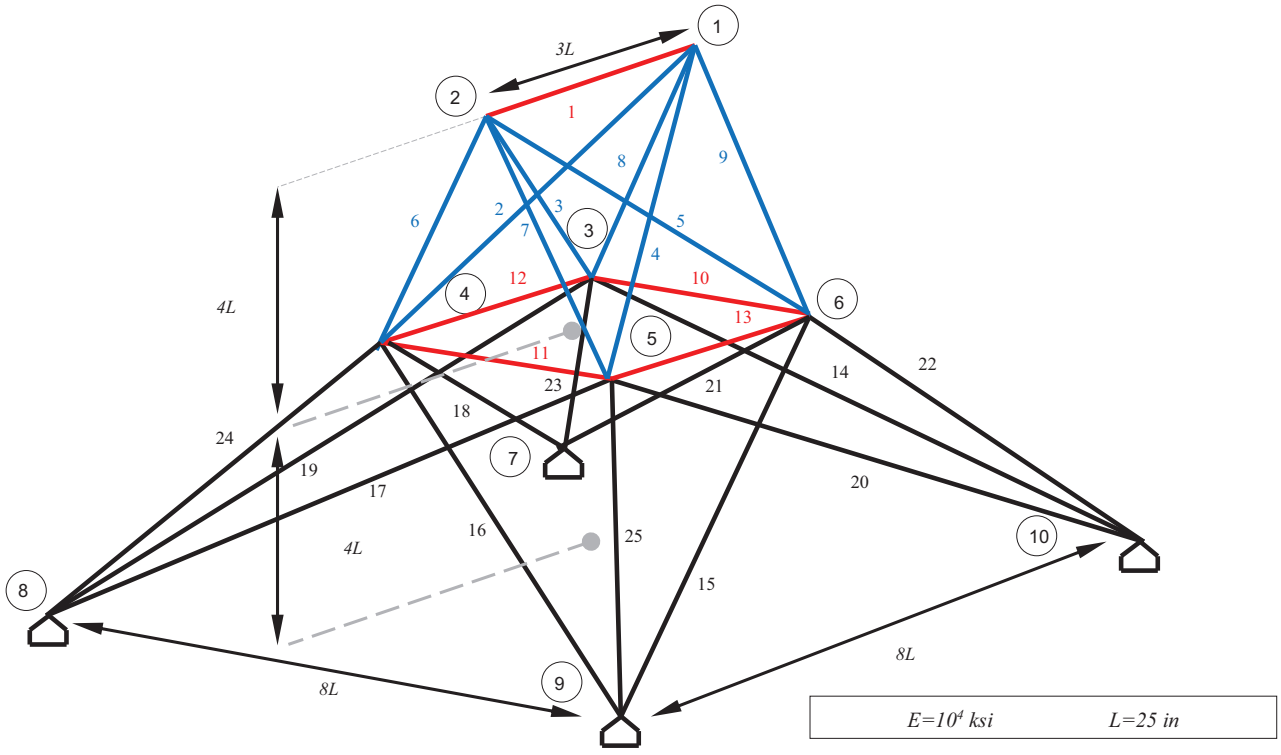

Fig. 13. Structure of a 25-bar truss.

Table 15
Member stress limitations for the 25-bar truss design problem.

Element group	Compressive stress limitations ksi (MPa)	Tensile stress limitations ksi (MPa)
Group 1 : A_1	35.092 (241.96)	40.0 (275.80)
Group 2 : A_2, A_3, A_4, A_5	11.590 (79.913)	40.0 (275.80)
Group 3 : A_6, A_7, A_8, A_9	17.305 (119.31)	40.0 (275.80)
Group 4 : A_{10}, A_{11}	35.092 (241.96)	40.0 (275.80)
Group 5 : A_{12}, A_{13}	35.092 (241.96)	40.0 (275.80)
Group 6 : A_{14}, A_{15}, A_{17}	6.759 (46.603)	40.0 (275.80)
Group 7 : $A_{18}, A_{19}, A_{20}, A_{21}$	6.959 (47.982)	40.0 (275.80)
Group 8 : $A_{22}, A_{23}, A_{24}, A_{25}$	11.082 (76.410)	40.0 (275.80)

- $E = 68,947$ MPa $(10,000$ ksi$)$
- Displacement limitation $= 0.35$ in.
- Maximum displacement $= 0.3504$ in.
- Design variabe set $= \left\{ \begin{matrix} 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8, \\ 0.9,\ 1.0,\ 1.1, 1.2,\ 1.3,\ 1.4,\ 1.5, \\ 1.6,\ 1.7,\ 1.8,\ 1.9,\ 2.0,\ 2.1,\ 2.2,\ 2.3, \\ 2.4,\ 2.6,\ 2.8,\ 3.0, 3.2,\ 3.4 \end{matrix} \right\}$

Member stress limitations for this truss are shown in Table 15. The 25-bar truss is subject to two leading conditions as listed in Table 16.

This problem has been solved in the literature in both binary [95,96,99–102] and continuous [103–108] version. We solve the continuous version of this problem to have a different problem than that of the previous and next subsections. We solve this problem using 30 search agents over 500 iterations and present the results in Table 17.

Table 17 shows that the best optimal weight obtained by WOA is 544.608 which is better than other algorithms. The results of average and standard deviation also show that WOA outperforms all the other algorithms except CSP. However, WOA requires substantially less number of analyses than CSP to find the best optimal design.

4.6. 52-bar truss design

This problem is another popular truss design problem with discrete variables. As may be seen in Fig. 14, there are 52 bars and 20 nodes of which four are fixed. The truss has 52 bars, which are classified in the following 12 groups:

- *Group* 1 : $A_1,\ A_2,\ A_3,\ A_4$
- *Group* 2 : $A_5,\ A_6,\ A_7,\ A_8,\ A_9,\ A_{10}$
- *Group* 3 : $A_{11},\ A_{12},\ A_{13}$
- *Group* 4 : $A_{14}, A_{15},\ A_{16},\ A_{17}$
- *Group* 5 : $A_{18},\ A_{19}, A_{20}, A_{21}, A_{22}, A_{23}$
- *Group* 6 : A_{24}, A_{25}, A_{26}
- *Group* 7 : $A_{27},\ A_{28}, A_{29},\ A_{30}$
- *Group* 8 : $A_{31}, A_{32}, A_{33}, A_{34}, A_{35}, A_{36}$
- *Group* 9 : A_{37}, A_{38}, A_{39}

Table 16
Two loading conditions for the 25-bar truss design problem.

Node	Case 1			Case 2		
	P_x kips (kN)	P_y kips (kN)	P_z kips (kN)	P_x kips (kN)	P_y kips (kN)	P_z kips (kN)
1	0.0	20.0 (89)	−5.0 (22.25)	1.0(4.45)	10.0 (44.5)	−5.0 (22.25)
2	0.0	−20.0 (89)	−5.0 (22.25)	0.0	10.0 (44.5)	−5.0 (22.25)
3	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0
6	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0

Table 17
Statistical comparison of WOA optimization results with literature for the 25-bar truss design problem.

Element group	Results of algorithms						
	ACO [103]	BB-BC [105]	PSO [108]	HPSACO [106]	CSS [107]	CSP [104]	WOA
A1	0.01	0.01	0.01	0.01	0.01	0.01	0.01
A2–A5	2.042	1.993	2.052	2.054	2.003	1.91	2.053
A6–A9	3.001	3.056	3.001	3.008	3.007	2.798	3.004
A10–A11	0.01	0.01	0.01	0.01	0.01	0.01	0.01
A12–A13	0.01	0.01	0.01	0.01	0.01	0.01	0.01
A14–A17	0.684	0.665	0.684	0.679	0.687	0.708	0.68
A18–A21	1.625	1.642	1.616	1.611	1.655	1.836	1.61
A22–A25	2.672	2.679	2.673	2.678	2.66	2.645	2.675
Best weight (lb)	545.03	545.16	545.21	544.99	545.10	545.09	544.608
Average weight (lb)	545.74	545.66	546.84	545.52	545.58	545.20	544.773
Std. dev. (lb)	0.94	0.491	1.478	0.315	0.412	0.487	0.498
No. of analyses	3520	12500	9596	9875	7000	17,500	9450

Table 18
Available cross-section areas of the AISC norm (valid values for the parameters).

No.	in.^{2}	mm^2	No.	in.^{2}	mm^2
1	0.111	71.613	33	3.84	2477.414
2	0.141	90.968	34	3.87	2496.769
3	0.196	126.451	35	3.88	2503.221
4	0.25	161.29	36	4.18	2696.769
5	0.307	198.064	37	4.22	2722.575
6	0.391	252.258	38	4.49	2896.768
7	0.442	285.161	39	4.59	2961.284
8	0.563	363.225	40	4.8	3096.768
9	0.602	388.386	41	4.97	3206.445
10	0.766	494.193	42	5.12	3303.219
11	0.785	506.451	43	5.74	3703.218
12	0.994	641.289	44	7.22	4658.055
13	1	645.16	45	7.97	5141.925
14	1.228	792.256	46	8.53	5503.215
15	1.266	816.773	47	9.3	5999.988
16	1.457	939.998	48	10.85	6999.986
17	1.563	1008.385	49	11.5	7419.34
18	1.62	1045.159	50	13.5	8709.66
19	1.8	1161.288	51	13.9	8967.724
20	1.99	1283.868	52	14.2	9161.272
21	2.13	1374.191	53	15.5	9999.98
22	2.38	1535.481	54	16	10322.56
23	2.62	1690.319	55	16.9	10903.2
24	2.63	1696.771	56	18.8	12129.01
25	2.88	1858.061	57	19.9	12838.68
26	2.93	1890.319	58	22	14193.52
27	3.09	1993.544	59	22.9	14774.16
28	3.13	2019.351	60	24.5	15806.42
29	3.38	2180.641	61	26.5	17096.74
30	3.47	2238.705	62	28	18064.48
31	3.55	2290.318	63	30	19354.8
32	3.63	2341.931	64	33.5	21612.86

- *Group* 10 : $A_{40},\ A_{41}, A_{42}, A_{43}$
- *Group* 11 : $A_{44}, A_{45}, A_{46}, A_{47}, A_{48}, A_{49}$
- *Group* 12 : A_{50}, A_{51}, A_{52}

Therefore, this problem has 12 parameters to be optimized. Other assumptions for this problem are as follows:

- $\rho = 7860.0\,\text{kg/m}^3$
- $E = 2.07e5\,\text{MPa}$
- Stress limitation = 180 MPa
- Maximum stress = 179.7652 MPa
- Design variable set is chosen from Table 18
- $P_k = 100\,\text{kN},\ P_y = 200\,\text{kN}$

We again employ 30 search agents over 500 iterations for solving this problem. Similarly to 15-bar truss design, the search agents of WOA were simply rounded to the nearest integer number during optimization since this problem is a discrete problem. The results are summarized and compared to several algorithms in the literature in Table 19.

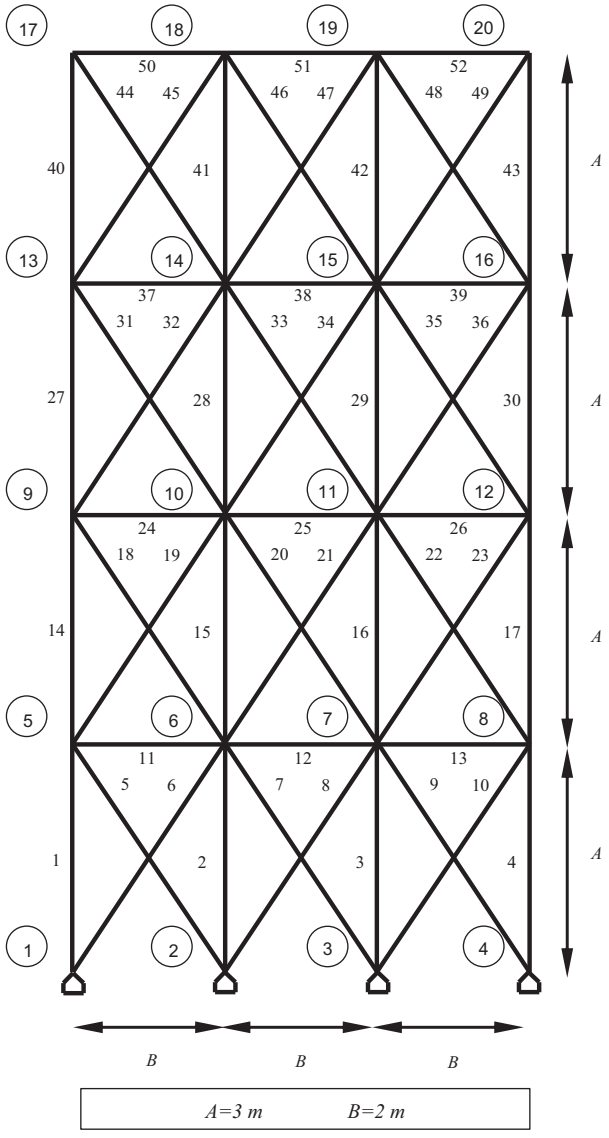

Fig. 14. Structure of a 52-bar truss.

According to Table 19, inspecting the results in this table, the best optimal weight obtained by WOA is 1902.605 which is identical to the optimal weights found by SOS and MBA. Also, the WOA

Table 19
Comparison of WOA optimization results with literature for the 52-bar truss design problem.

Variables (mm^2)	PSO [95]	PSOPC [95]	HPSO [95]	DHPSACO [109]	MBA [96]	SOS [98]	WOA
A1–A4	4658.055	5999.988	4658.055	4658.055	4658.055	4658.055	4658.055
A5–A10	1374.19	1008.38	1161.288	1161.288	1161.288	1161.288	1161.288
A11–A13	1858.06	2696.77	363.225	494.193	494.193	494.193	494.193
A14–A17	3206.44	3206.44	3303.219	3303.219	3303.219	3303.219	3303.219
A18–A23	1283.87	1161.29	940	1008.385	940	940	940
A24–A26	252.26	729.03	494.193	285.161	494.193	494.193	494.193
A27–A30	3303.22	2238.71	2238.705	2290.318	2238.705	2238.705	2238.705
A31–A36	1045.16	1008.38	1008.385	1008.385	1008.385	1008.385	1008.385
A37–A39	126.45	494.19	388.386	388.386	494.193	494.193	494.193
A40–A43	2341.93	1283.87	1283.868	1283.868	1283.868	1283.868	1283.868
A44–A49	1008.38	1161.29	1161.288	1161.288	1161.288	1161.288	1161.288
A50–A52	1045.16	494.19	792.256	506.451	494.193	494.193	494.193
Optimal weight (kg)	2230.16	2146.63	1905.495	1904.83	1902.605	1902.605	1902.605
No. of analyses	150,000	150,000	5300	11,100	5450	2350	2250

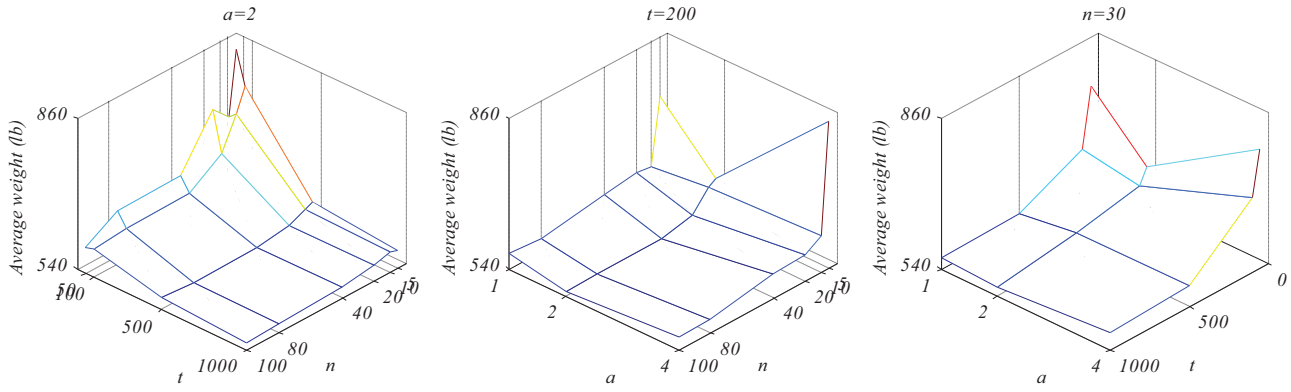

Fig. 15. Average weight obtained by WOA over 10 runs for the 52-bar truss varying *n*, *t*, and *a*.

requires the least number of function evaluations when solving this problem. It is evident from the results that WOA, SOS, and MBA significantly outperformed PSO, PSOPC, HPSO, and DHPSACO.

Since the WOA algorithm is a novel optimization paradigm, the study of the main internal parameters of this algorithm such as number of search agents, maximum iteration number, and the vector a in Eq. (2.3) would be very helpful for the researchers who is going to apply this algorithm to different problems. Therefore, we solve the 52-bar truss design problem with varying these parameters as follows:

- Number of search agents (n): 5, 10, 20, 40, or 80, 100
- Maximum iteration number (t): 50, 100, 500, or 1000
- Vector a: linearly decreases from 1 to 0, 2 to 0, or 4 to 0

To be able to illustrate the effect of these parameters on the performance of the WOA algorithm, three independent experiments are done by simultaneously varying n and t, n and a, or t and a. The 52-bar truss problem is solved 54 ($6*4+6*3+4*3$) rounds by WOA with different values for n, t, and a. For every WOA with different parameters, we have run it 10 times on the problem to be able to calculate the average weight and reliably compare the results. After all, the results are illustrated in Fig. 15.

This figure shows that that the best values for n and t are equal to 100 and 1000, respectively. These results are reasonable because the larger number of search agents and maximum iterations, the better approximation of the global optimum. For the parameter a, it seems that the linear decrement from 2 to 0 results in better results. Other values give worse results because they either merely/less emphasize exploitation or exploitation.

As summary, the results of the structural design problems revealed that the proposed WOA algorithm has the potential to be very effective in solving real problems with unknown search spaces as well.

5. Conclusion

This study presented a new swarm-based optimization algorithm inspired by the hunting behavior of humpback whales. The proposed method (named as WOA, Whale Optimization Algorithm) included three operators to simulate the search for prey, encircling prey, and bubble-net foraging behavior of humpback whales. An extensive study was conducted on 29 mathematical benchmark functions to analyze exploration, exploitation, local optima avoidance, and convergence behavior of the proposed algorithm. WOA was found to be enough competitive with other state-of-the-art meta-heuristic methods.

In order to gather further information, six structural engineering problems (*i.e.* design of a tension/compression spring, design of a welded beam, design of a pressure vessel, design of a 15-bar truss, design of a 25-bar truss, and design of a 52-bar truss design) were solved. WOA was very competitive with meta-heuristic optimizers and superior over conventional techniques.

Binary and multi-objective versions of the WOA algorithm called, respectively, Binary Whale Optimization Algorithm and Multi-Objective Whale Optimization Algorithm are currently under development.

Acknowledgment

We would like to acknowledge Prof. A. Kaveh's publications as one of the pioneers in the field of stochastic optimization that always have been an inspiration to us.

References

[1] Holland JH. Genetic algorithms. Sci Am 1992;267:66–72.
[2] J.R. Koza, "Genetic programming," 1992.
[3] Simon D. Biogeography-based optimization. IEEE Trans Evol Comput 2008;12:702–13.
[4] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simmulated annealing. Science 1983;220:671–80.
[5] Černý V. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J Opt Theory Appl 1985;45:41–51.
[6] Webster B, Bernhard PJ. A local search optimization algorithm based on natural principles of gravitation. In: Proceedings of the 2003 international conference on information and knowledge engineering (IKE'03); 2003. p. 255–61.
[7] Erol OK, Eksin I. A new optimization method: big bang–big crunch. Adv Eng Softw 2006;37:106–11.
[8] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm. Inf Sci 2009;179:2232–48.
[9] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mech 2010;213:267–89.
[10] Formato RA. Central force optimization: A new metaheuristic with applications in applied electromagnetics. Prog Electromag Res 2007;77:425–91.
[11] Alatas B. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. Expert Syst Appl 2011;38:13170–80.
[12] Hatamlou A. Black hole: a new heuristic optimization approach for data clustering. Inf Sci 2013;222:175–84.
[13] Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization. Comput Struct 2012;112:283–94.
[14] Du H, Wu X, Zhuang J. Small-world optimization algorithm for function optimization. Advances in natural computation. Springer; 2006. p. 264–73.
[15] Shah-Hosseini H. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int J Comput Sci Eng 2011;6:132–40.
[16] Moghaddam FF, Moghaddam RF, Cheriet M. Curved space optimization: A random search based on general relativity theory. 2012. arXiv:1208.2214.
[17] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the 1995 IEEE international conference on neural networks; 1995. p. 1942–8.
[18] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Comput Intell 2006;1:28–39.
[19] Abbass HA. MBO: Marriage in honey bees optimization – a haplometrosis polygynous swarming approach. In: Proceedings of the 2001 congress on evolutionary computation; 2001. p. 207–14.
[20] Li X. A new intelligent optimization-artificial fish swarm algorithm [Doctor thesis]. China: Zhejiang University of Zhejiang; 2003.
[21] Roth M, Stephen W. Termite: A swarm intelligent routing algorithm for mobilewireless Ad-Hoc networks. Stigmergic Optimization. Springer Berlin Heidelberg; 2006. p. 155–84.
[22] Basturk B, Karaboga D. An artificial bee colony (ABC) algorithm for numeric function optimization. In: Proceedings of the IEEE swarm intelligence symposium; 2006. p. 12–14.
[23] Pinto PC, Runkler TA, Sousa JM. Wasp swarm algorithm for dynamic MAX-SAT problems. Adaptive and natural computing algorithms. Springer; 2007. p. 350–7.
[24] Mucherino A, Seref O. Monkey search: a novel metaheuristic search for global optimization. In: AIP conference proceedings; 2007. p. 162.
[25] Yang C, Tu X, Chen J. Algorithm of marriage in honey bees optimization based on the wolf pack search. In: Proceedings of the 2007 international conference on intelligent pervasive computing, IPC; 2007. p. 462–7.
[26] Lu X, Zhou Y. A novel global convergence algorithm: bee collecting pollen algorithm. Advanced intelligent computing theories and applications With aspects of artificial intelligence. Springer; 2008. p. 518–25.
[27] Yang X-S, Deb S. Cuckoo search via Lévy flights. In: Proceedings of the world congress on nature & biologically inspired computing, NaBIC 2009; 2009. p. 210–14.
[28] Shiqin Y, Jianjun J, Guangxing Y. A dolphin partner optimization. In: Proceedings of the WRI global congress on intelligent systems, GCIS'09; 2009. p. 124–8.
[29] Yang X-S. A new metaheuristic bat-inspired algorithm. In: Proceedings of the workshop on nature inspired cooperative strategies for optimization (NICSO 2010). Springer; 2010. p. 65–74.
[30] Yang X-S. Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2010;2:78–84.
[31] Oftadeh R, Mahjoob MJ, Shariatpanahi M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search. Comput Math Appl 2010;60:2087–98.
[32] Askarzadeh A, Rezazadeh A. A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: bird mating optimizer. Int J Energy Res 2012.
[33] Gandomi AH, Alavi AH. Krill Herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 2012;17(12):4831–45.
[34] Pan W-T. A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowledge-Based Syst 2012;26:69–74.
[35] Kaveh A, Farhoudi N. A new optimization method: dolphin echolocation. Adv Eng Softw 2013;59:53–70.
[36] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf Sci 2012;183:1–15.
[37] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-Aided Des 2011;43:303–15.
[38] Geem ZW, Kim JH, Loganathan G. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60–8.
[39] Fogel D. Artificial intelligence through simulated evolution. Wiley-IEEE Press; 2009.
[40] Glover F. Tabu search – Part I. ORSA J Comput 1989;1:190–206.
[41] Glover F. Tabu search – Part II. ORSA J Comput 1990;2:4–32.
[42] He S, Wu Q, Saunders J. A novel group search optimizer inspired by animal behavioural ecology. In: Proceedings of the 2006 IEEE congress on evolutionary computation, CEC; 2006. p. 1272–8.
[43] He S, Wu QH, Saunders J. Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 2009;13:973–90.
[44] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Proceedings of the 2007 IEEE congress on evolutionary computation, CEC; 2007. p. 4661–7.
[45] Kashan AH. League championship algorithm: a new algorithm for numerical function optimization. In: Proceedings of the international conference on soft computing and pattern recognition, SOCPAR'09.; 2009. p. 43–8.
[46] Husseinzadeh Kashan A. An efficient algorithm for constrained global optimization and application to mechanical engineering design: league championship algorithm (LCA). Computer-Aided Des 2011;43:1769–92.
[47] Tan Y, Zhu Y. Fireworks algorithm for optimization. Advances in swarm intelligence. Springer; 2010. p. 355–64.
[48] Kaveh A. Colliding bodies optimization. Advances in metaheuristic algorithms for optimal design of structures. Springer; 2014. p. 195–232.
[49] Kaveh A, Mahdavi V. Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 2014;139:18–27.
[50] Gandomi AH. Interior search algorithm (ISA): a novel approach for global optimization. ISA Trans 2014.
[51] 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.
[52] Moosavian N, Roodsari BK. Soccer league competition algorithm: a new method for solving systems of nonlinear equations. Int J Intell Sci 2013;4:7.
[53] Moosavian N, Kasaee Roodsari B. Soccer league competition algorithm: a novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evol Comput 2014;17:14–24.
[54] Dai C, Zhu Y, Chen W. Seeker optimization algorithm. Computational intelligence and security. Springer; 2007. p. 167–76.
[55] Ramezani F, Lotfi S. Social-based algorithm (SBA). Appl Soft Comput 2013;13:2837–56.
[56] Ghorbani N, Babaei E. Exchange market algorithm. Appl Soft Comput 2014;19:177–87.
[57] Eita MA, Fahmy MM. Group counseling optimization. Appl Soft Comput 2014;22:585–604.
[58] Eita MA, Fahmy MM. Group counseling optimization: a novel approach. In: Bramer M, Ellis R, Petridis M, editors. Research and development in intelligent systems XXVI. London: Springer; 2010. p. 195–208.
[59] Olorunda O, Engelbrecht AP. Measuring exploration/exploitation in particle swarms using swarm diversity. In: Proceedings of the 2008 IEEE congress on evolutionary computation, CEC (IEEE world congress on computational intelligence); 2008. p. 1128–34.
[60] Alba E, Dorronsoro B. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans Evol Comput 2005;9:126–42.
[61] Lin L, Gen M. Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation. Soft Comput 2009;13:157–68.
[62] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61.
[63] Hof PR, Van Der Gucht E. Structure of the cerebral cortex of the humpback whale, *Megaptera novaeangliae* (Cetacea, Mysticeti, Balaenopteridae). Anat Rec 2007;290:1–31.
[64] Watkins WA, Schevill WE. Aerial observation of feeding behavior in four baleen whales: *Eubalaena glacialis*, *Balaenoptera borealis*, *Megaptera novaeangliae*, and *Balaenoptera physalus*. J Mammal 1979:155–63.
[65] Goldbogen JA, Friedlaender AS, Calambokidis J, Mckenna MF, Simon M, Nowacek DP. Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology. BioScience 2013;63:90–100.
[66] Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans Evol Comput 1999;3:82–102.
[67] Digalakis J, Margaritis K. On benchmarking functions for genetic algorithms. Int J Comput Math 2001;77:481–506.
[68] Molga M, Smutnicki C. Test functions for optimization needs; 2005. http://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf.
[69] Yang X-S. Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2010;2(2):78–84.

[70] Liang J, Suganthan P, Deb K. Novel composition test functions for numerical global optimization. In: Proceedings of the 2005 swarm intelligence symposium, SIS; 2005. p. 68–75.
[71] 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, 2005005, 2005.
[72] Salomon R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems 1996;39:263–78.
[73] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 1997;11:341–59.
[74] Hansen N, Müller SD, Koumoutsakos P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 2003;11:1–18.
[75] van den Bergh F, Engelbrecht A. A study of particle swarm optimization particle trajectories. Inf Sci 2006;176:937–71.
[76] 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.
[77] Arora JS. Introduction to optimum design. Academic Press; 2004.
[78] Belegundu AD. Study of mathematical programming methods for structural optimization. Diss Abstr Int Part B: Sci Eng 1983;43:1983.
[79] Coello Coello CA, Mezura Montes E. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 2002;16:193–203.
[80] He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 2007;20:89–99.
[81] Mezura-Montes E, Coello Coello CA. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 2008;37:443–73.
[82] Coello Coello CA. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 2000;41:113–27.
[83] Mahdavi M, Fesanghary M, Damangir E. An improved harmony search algorithm for solving optimization problems. Appl Math Comput 2007;188:1567–79.
[84] Li L, Huang Z, Liu F, Wu Q. A heuristic particle swarm optimizer for optimization of pin connected structures. Comput Struct 2007;85:340–9.
[85] Yang XS. Nature-inspired metaheuristic algorithms. Luniver Press; 2011.
[86] Carlos A, Coello C. Constraint-handling using an evolutionary multiobjective optimization technique. Civil Eng Syst 2000;17:319–46.
[87] Deb K. Optimal design of a welded beam via genetic algorithms. AIAA J 1991;29:2013–15.
[88] Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods Appl Mech Eng 2000;186:311–38.
[89] Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods Appl Mech Eng 2005;194:3902–33.
[90] Ragsdell K, Phillips D. Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind 1976;98:1021–5.
[91] Deb K. GeneAS: A robust optimal design technique for mechanical component design. Evolutionary algorithms in engineering applications. Springer Berlin Heidelberg; 1997. p. 497–514.
[92] Kaveh A, Talatahari S. An improved ant colony optimization for constrained engineering design problems. Eng Comput: Int J Computer-Aided Eng 2010;27:155–82.
[93] 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.
[94] Sandgren E. Nonlinear integer and discrete programming in mechanical design optimization. J Mech Design 1990;112(2):223–9.
[95] Li L, Huang Z, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables. Comput Struct 2009;87:435–43.
[96] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm for optimization of truss structures with discrete variables. Comput Struct 2012;102:49–63.
[97] Zhang Y, Liu J, Liu B, Zhu C, Li Y. Application of improved hybrid genetic algorithm to optimize. J South China Univ Technol 2003;33:69–72.
[98] Cheng M-Y, Prayogo D. Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 2014;139:98–112.
[99] Wu S-J, Chow P-T. Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 1995;56:979–91.
[100] Rajeev S, Krishnamoorthy C. Discrete optimization of structures using genetic algorithms. J Struct Eng 1992;118:1233–50.
[101] Lee KS, Geem ZW, Lee S-h, Bae K-w. The harmony search heuristic algorithm for discrete structural optimization. Eng Optim 2005;37:663–84.
[102] Ringertz UT. On methods for discrete structural optimization. Eng Optim 1988;13:47–64.
[103] Camp CV, Bichon BJ. Design of space trusses using ant colony optimization. J Struct Eng 2004;130:741–51.
[104] 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.
[105] Kaveh A, Talatahari S. Size optimization of space trusses using Big Bang–Big Crunch algorithm. Comput Struct 2009;87:1129–40.
[106] Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 2009;87:267–83.
[107] Kaveh A, Talatahari S. Optimal design of skeletal structures via the charged system search algorithm. Struct Multidiscip Optim 2010;41:893–911.
[108] Schutte J, Groenwold A. Sizing design of truss structures using particle swarms. Struct Multidiscip Optim 2003;25:261–9.
[109] Kaveh A, Talatahari S. A particle swarm ant colony optimization for truss structures with discrete variables. J Constr Steel Res 2009;65:1558–68.
[110] Rechenberg I. Evolutionsstrategien. Springer Berlin Heidelberg; 1978. p. 83–114.
[111] Dasgupta D, Zbigniew M, editors. Evolutionary algorithms in engineering applications. Springer Science & Business Media; 2013.