Engineering Applications of Artificial Intelligence 87 (2020) 103300

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications

Weiguo Zhao [a,b], Zhenxing Zhang [b], Liying Wang [a,*]

[a] School of Water Conservancy and Hydropower, Hebei University of Engineering, Handan, Hebei, 056021, China
[b] Illinois State Water Survey, Prairie Research Institute, University of Illinois at Urbana-Champaign, Champaign, IL 61820, USA

ARTICLE INFO

Keywords:
Global optimization
Engineering design
Manta ray foraging optimization
Metaheuristic
Constrained problems
Optimization algorithm
Heuristic algorithm

ABSTRACT

A new bio-inspired optimization technique, named Manta Ray Foraging Optimization (MRFO) algorithm, is proposed and presented, aiming to providing a novel algorithm that provides an alternate optimization approach for addressing real-world engineering issues. The inspiration of this algorithm is based on intelligent behaviors of manta rays. This work mimics three unique foraging strategies of manta rays, including chain foraging, cyclone foraging, and somersault foraging, to develop an efficient optimization paradigm for solving different optimization problems. The performance of MRFO is evaluated, through comparisons with other state-of-the-art optimizers, on benchmark optimization functions and eight real-world engineering design cases. The comparison results on the benchmark functions suggest that MRFO is far superior to its competitors. In addition, the real-world engineering applications show the merits of this algorithm in tackling challenging problems in terms of computational cost and solution precision. The MATLAB codes of the MRFO algorithm are available at https://www.mathworks.com/matlabcentral/fileexchange/73130-manta-ray-foraging-optimization-mrfo.

1. Introduction

Many real-world optimization problems are increasingly becoming challenging as they often concern a big number of decision variables, complex nonlinear constraints and objective functions. The global optimization using traditional approaches like numerical methods becomes less powerful especially when objective functions or constraints have multiple peaks. Metaheuristic algorithms, powerful tools for handling challenging optimization problems, are increasingly becoming popular. The popularity drives from the following aspects.

First, the most outstanding characteristic of metaheuristic algorithms is their simplicity. These metaheuristic methods possess basic theories or mathematical models which derive from nature. These methods are generally simple and easy to implement. The ease-to-use allows one to apply metaheuristics to solve real-world problems. Moreover, it is also easy to develop their variants according to existing methods. Second, these optimization technologies can be viewed as a black box, meaning that it is able to offer a set of outputs for a given problem for a set of inputs. Furthermore, scholars may easily modify the structures and parameters of these methods in order to obtain satisfactory solutions. Third, randomness is one the most important characteristics of metaheuristic algorithms. This allows metaheuristic algorithms to explore the entire search space and prevent them from trapping into local optima effectively. More specially, it makes many metaheuristics successful to solve problems with unknown search space or multiple local optima. Finally, these metaheuristics are highly versatile and flexible, implying their practicability to various different types of optimization problems including non-linear problems, non-differentiable problems, or complex numerical problems with plentiful of local minima. Many metaheuristic algorithms have been presented and successfully applied to different areas. These algorithms are mainly categorized into three classes (Hare et al., 2013): evolution-based (Mühlenbein et al., 1988), physics-based (Geem et al., 2001), and swarm-based (Krause et al., 2013).

Evolution-based algorithms simulate natural evolution like chemotaxis, reproduction elimination and dispersal, and migration (Passino, 2002; De Falco et al., 2012). Genetic algorithm (GA), proposed by Holland (1975), is a famous and widely used evolutionary algorithm (EA). One of the main features of GA is that it does not require derivative in the search space that is existing in mathematical optimization approaches. GA evolves a population by emulating survival of the fittest in nature, it may provide efficient solutions and avoid local optima. Since its emergence, a range of variants have been developed to improve GA. With its popularity, many other evolutionary algorithms, including differential evolution (DE) (Rocca et al., 2011), evolutionary programming (EP) (Juste et al., 1999), evolutionary strategies (ES) (Beyer and Schwefel, 2002), memetic algorithm (MA) (Moscato et al., 2007), and so on, have been proposed. In addition, scores of novel

* Corresponding author.
E-mail addresses: zhaoweiguo@hebeu.edu.cn (W. Zhao), zhang538@illinois.edu (Z. Zhang), wangliying@hebeu.edu.cn (L. Wang).

https://doi.org/10.1016/j.engappai.2019.103300
Received 21 December 2018; Received in revised form 17 September 2019; Accepted 14 October 2019
Available online xxxx
0952-1976/

EAs have been presented recently, including biogeography-based optimization (BBO) (Simon, 2009), bacterial foraging optimization (BFO) (Passino, 2002), artificial algae algorithm (AAA) (Uymaz et al., 2015), bat algorithm (BA) (Yang and Hossein Gandomi, 2012), monkey king evolutionary (MKE) (Meng and Pan, 2016), et al. (Punnathanam and Kotecha, 2016; Pan, 2012; Mehrabian and Lucas, 2006; Civicioglu, 2013).

Physics-based algorithms mimic physical laws in universe. Simulated annealing (SA) (Kirkpatrick et al., 1983) is one of the most well-known physics-based techniques. SA is analogy with thermodynamics in physical material. Annealing is to minimize energy use, specifically with the way that heated metals cool and crystallize. Recently, multiple new physics-inspired techniques are developed, including gravitational search algorithm (GSA) (Rashedi et al., 2009), electromagnetism-like mechanism (EM) algorithm (Birbil and Fang, 2003), particle collision algorithm (PCA) (Sacco and De Oliveira, 2005), vortex search algorithm (VSA) (Doğan and Ölmez, 2015), water evaporation optimization (WEO) (Kaveh and Bakhshpoori, 2016), atom search optimization (ASO) (Zhao et al., 2019a), big bang–big crunch algorithm (BB-BC) (Genç et al., 2010), et al. (Shah-Hosseini, 2011; Chuang and Jiang, 2007; Shah-Hosseini, 2009; Kaveh and Dadras, 2017; Kaveh and Talatahari, 2010; Zheng et al., 2010; Javidy et al., 2015; Mirjalili and Hashim, 2012; Zheng, 2015; Flores et al., 2011; Tamura and Yasuda, 2011; Rao et al., 2012; Zarand et al., 2002; Shen and Li, 2009; Kripka and Kripka, 2008; Patel and Savsani, 2015; Eskandar et al., 2012; Moghaddam et al., 2012).

Swarm-based algorithms simulate social behaviors of species like self-organization and division of labor (Beni and Wang, 1993; Ab Wahab et al., 2015). Two outstanding examples are particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) and ant colony optimization (ACO) (Dorigo et al., 1996). PSO inspired by bird flocking behaviors updates each agent in a population by its best individual agent and the best global agent. ACO is inspired by the foraging behaviors of ant swarms, ants search for the most effective route from their nest to the food source by the intensity of pheromones which reduces over time. Other examples of swarm-inspired algorithms include glowworm swarm optimization (GSO) (Krihnanand and Ghose, 2009), grey wolf optimization (GWO) (Mirjalili et al., 2014), artificial ecosystem-based optimization (AEO) (Zhao et al., 2019b), shark smell optimization (SSO) (Abedinia et al., 2014), firefly algorithm (FA) (Yang, 2010), supply–demand-based optimization (SDO) (Zhao et al., 2019c), spotted hyena optimization (SHO) (Gaurav and Vijay, 2017), and so on (Yang and Deb, 2009; Oftadeh et al., 2010; Kiran, 2015; Mohamed et al., 2017; Cuevas et al., 2013; Askarzadeh, 2014; Saremi et al., 2017; Akay and Karaboga, 2012a,b; Mirjalili, 2016; Kaveh and Farhoudi, 2013; Yang, 2010; Mucherino and Seref, 2007; Gandomi and Alavi, 2012).

It is worth mentioning that there are other recently developed metaheuristics motivated from social behaviors and ideology in humans. Some of the most well-known ones include parliamentary optimization algorithm (POA) (Borji, 2007), artificial human optimization (AHO) (Gajawada, 2016), continuous opinion dynamics optimization (CODO) (Kaur et al., 2013), league championship algorithm (LCA) (Kashan, 2009), social group optimization (SGO) (Satapathy and Naik, 2016), ideology algorithm (IA) (Huan et al., 2016), and so on (Kuo and Lin, 2013; Moosavian and Roodsari, 2014; Kumar et al., 2018; Xu et al., 2010; Ray and Liew, 2003).

Swarm-based algorithms have some unique features. Some historical information about the swarm is restored to provide a basis for every agent to update individual positions by using interaction rules over subsequent iterations. In general, swarm-based techniques share two specific behaviors, exploration and exploitation (Alba and Dorronsoro, 2005; Lynn and Suganthan, 2015). Exploration is to search a wide variable space for promising solutions which are not neighbor to the current solution, and this search should be as extensive and random as possible. This behavior generally contributes to escaping local optima. Exploitation is to confine the search to a small region found in the exploration process to refine the solution. Essentially this behavior implements local search in a promising space. Based on these two behaviors, swarm-based algorithms have superiority over other two types of algorithms.

Some might question why new optimization algorithms are still developed despite of so many existing algorithms. The answer can be found in the No Free Lunch Theorem of Optimization (Wolpert and Macready, 1997), which describes that there is no optimizer performing the best for all optimization problems. So developing an effective swarm-inspired optimizer to solve specific real-world problems motivates this work.

This paper proposes a new metaheuristic algorithm, named manta ray foraging optimization (MRFO), which simulates the foraging behaviors of manta rays. This algorithm has three foraging operators, including chain foraging, cyclone foraging, and somersault foraging. The performance of MRFO is tested using 31 test functions and 8 engineering problems. The test results discover that the proposed method significantly outperforms those well-known metaheuristics.

This study is structured as follows. Section 2 detailedly introduces MRFO algorithm and the concepts behind it. 31 mathematical optimization problems and 8 real-world engineering problems are employed to check the validity of MRFO in Sections 3 and 4, respectively. Finally, Section 5 gives some conclusions and suggests future research directions.

2. Manta Ray Foraging Optimization (MRFO)

2.1. Inspiration

Manta rays are fancy creatures although they appear to be terrible. They are one of the largest known marine creatures. Manta rays have a flat body from top to bottom and a pair of pectoral fins, with which they elegantly swim as birds freely fly. They also have a pair of cephalic lobes that extend in front of their giant, terminal mouths. Fig. 1(A) provided by Swanson Chan on Unsplash depicts a foraging manta ray, and Fig. 1(B) shows the structure of a manta ray. Without sharp teeth, manta rays feed on plankton made mostly of microscopic animals from the water. When foraging, they funnel water and prey into their mouths using horn-shaped cephalic lobes. Then the prey is filtered from the water by modified gill rakers (Dewar et al., 2008). Manta rays can be divided into two distinct species. One is reef manta rays (manta alfredi) living in Indian Ocean and western and south Pacific, which can reach 5.5 m in width. The other is giant manta rays (manta birostris) found throughout tropical, subtropical and warm temperate oceans, which can reach 7 m in width. They have existed for about 5 million years. The average life span is 20 years but many never reach this age because they are hunted by fishers (Miller and Klimovich, 2016).

Manta rays eat a large amount of plankton every day. An adult manta ray can eat 5 kg of plankton on daily basis. Oceans are believed to be the richest source of plankton. However, plankton is not evenly dispersed or regularly concentrated in some specific areas, which are formed with ebb and flow of tides or change of seasons. Interestingly, manta rays are always good at finding abundant plankton. The most intriguing thing about manta rays is their foraging behaviors, and they may travel alone or in groups of up to 50 but foraging is often observed in groups. These creatures have evolved a variety of fantastic and intelligent foraging strategies.

The first foraging strategy is chain foraging (Johnna, 2016). When 50 or more manta rays start foraging, they line up, one behind another, forming an orderly line. Smaller male manta rays are piggybacked upon female ones and swim on top of their backs to match the beats of the female's pectoral fins. Consequently, the plankton which is missed by previous manta rays will be scooped up by ones behind them. Cooperating with each other in this way, they can funnel the most amount of plankton into their gills and improve their food rewards.

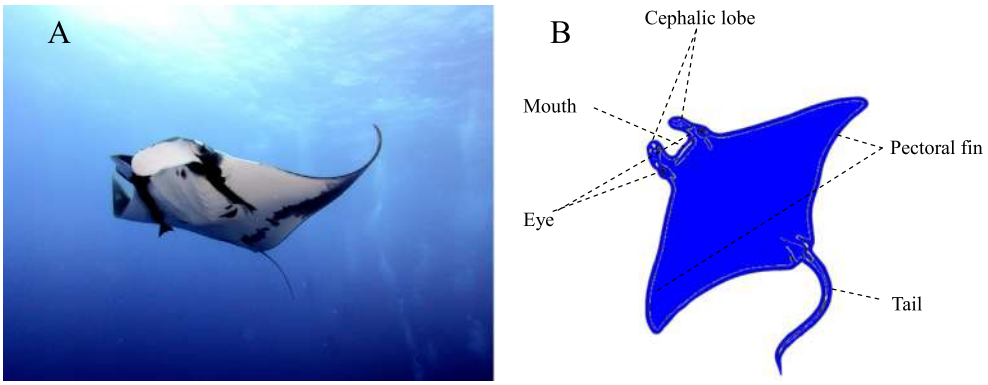

Fig. 1. (A) A foraging manta ray, and (B) structure of a manta ray.

The second foraging strategy is cyclone foraging (Gene and George, 2014). When the concentration of plankton is very high, dozens of manta rays gather together. Their tail ends link up with heads in a spiral to generate a spiraling vertex in the eye of the cyclone and the filtered water moves up towards the surface. This pulls the plankton into their open mouths.

The final foraging strategy is somersault foraging (Rebecca and Bobbie, 2015). This is one of the most splendid sceneries in nature. When manta rays find a food source, they will do a series of backwards somersaults, circling around the plankton to draw it towards manta rays. Somersault is a random, frequent, local and cyclical movement, which helps manta rays to optimize food intake.

Although these foraging behaviors are rare in nature, they are highly effective. We mathematically model these foraging behaviors and develop a new metaheuristic algorithm named Manta Ray Foraging Optimization (MRFO) to perform a global optimization.

2.2. Mathematical model

MRFO is inspired by three foraging behaviors including chain foraging, cyclone foraging and somersault foraging. The mathematical models are described below.

2.2.1. Chain foraging

In MRFO, manta rays can observe the position of plankton and swim towards it. The higher the concentration of plankton in a position is, the better the position is. Although the best solution is not known, MRFO assumes the best solution found so far is the plankton with high concentration manta rays want to approach and eat. Manta rays line up head-to-tail and form a foraging chain. Individuals except the first move towards not only the food but also the one in front of it. That is, in every iteration, each individual is updated by the best solution found so far and the solution in front of it. This mathematical model of chain foraging is represented as follows

$$x_i^d(t+1) = \begin{cases} x_i^d(t) + r \cdot (x_{best}^d(t) - x_i^d(t)) + \alpha \cdot (x_{best}^d(t) - x_i^d(t)) & i = 1 \\ x_i^d(t) + r \cdot (x_{i-1}^d(t) - x_i^d(t)) + \alpha \cdot (x_{best}^d(t) - x_i^d(t)) & i = 2, \ldots, N \end{cases} \tag{1}$$

$$\alpha = 2 \cdot r \cdot \sqrt{|log(r)|} \tag{2}$$

where, $x_i^d(t)$ is the position of ith individual at time t in dth dimension, r is a random vector within the range of [0, 1], a is a weight coefficient, $x_{best}^d(t)$ is the plankton with high concentration. Fig. 2 depicts this foraging behavior in a 2-D space. The position update of the ith individual is determined by the position $x_{i-1}(t)$ of the (i-1)th current individual and the position $x_{best}(t)$ of the food.

2.2.2. Cyclone foraging

When a school of manta rays recognize a patch of plankton in deep water, they will form a long foraging chain and swim towards the food by a spiral. This similar spiral foraging strategy can be found in WOA (Mirjalili and Lewis, 2016). However, for the cyclone foraging strategy of manta ray swarms, in addition to spirally move towards the food, each manta ray swims towards the one in front of it. That is, manta ray swarms in line developing a spiral perform foraging. Fig. 3 illustrates the cyclone foraging behavior in a 2-D space. An individual not only follows the one in front of it but only moves towards the food along a spiral path. The mathematical equation modeling the spiral-shaped movement of manta rays in a 2-D space can be defined as

$$\begin{cases} X_i(t+1) = X_{best} + r \cdot (X_{i-1}(t) - X_i(t)) + e^{bw} \cdot \cos(2\pi w) \cdot (X_{best} - X_i(t)) \\ Y_i(t+1) = Y_{best} + r \cdot (Y_{i-1}(t) - Y_i(t)) + e^{bw} \cdot \sin(2\pi w) \cdot (Y_{best} - Y_i(t)) \end{cases} \tag{3}$$

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

This motion behavior may be extended to a n-D space. For simplicity, this mathematical model of cyclone foraging can be defined as

$$x_i^d(t+1) = \begin{cases} x_{best}^d + r \cdot (x_{best}^d(t) - x_i^d(t)) + \beta \cdot (x_{best}^d(t) - x_i^d(t)) & i = 1 \\ x_{best}^d + r \cdot (x_{i-1}^d(t) - x_i^d(t)) + \beta \cdot (x_{best}^d(t) - x_i^d(t)) & i = 2, \ldots, N \end{cases} \tag{4}$$

$$\beta = 2e^{r_1 \frac{T-t+1}{T}} \cdot \sin(2\pi r_1) \tag{5}$$

where β is the weight coefficient, T is the maximum number of iterations, and r_1 is the rand number in [0, 1].

All individuals randomly perform the search with respect to the food as their reference position, so the cyclone foraging has a good exploitation for the region with the best solution found so far. This behavior is also used to substantially improve the exploration. We can force each individual to search for a new position far from the current best one by assigning a new random position in the entire search space as their reference position. This mechanism focuses mainly on the exploration and enables MRFO to achieve an extensive global search, its mathematical equation is presented below

$$x_{rand}^d = Lb^d + r \cdot (Ub^d - Lb^d) \tag{6}$$

$$x_i^d(t+1) = \begin{cases} x_{rand}^d + r \cdot (x_{rand}^d - x_i^d(t)) + \beta \cdot (x_{rand}^d - x_i^d(t)) & i = 1 \\ x_{rand}^d + r \cdot (x_{i-1}^d(t) - x_i^d(t)) + \beta \cdot (x_{rand}^d - x_i^d(t)) & i = 2, \ldots, N \end{cases} \tag{7}$$

where x_{rand}^d is a random position randomly produced in the search space, Lb^d and Ub^d are the lower and upper limits of the dth dimension, respectively.

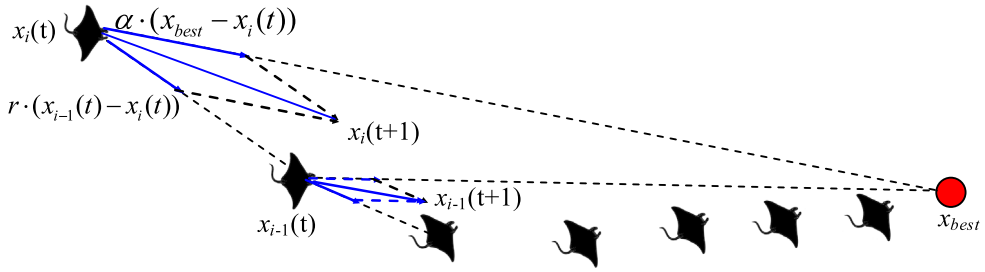

Fig. 2. Chain foraging behavior in a 2-D space.

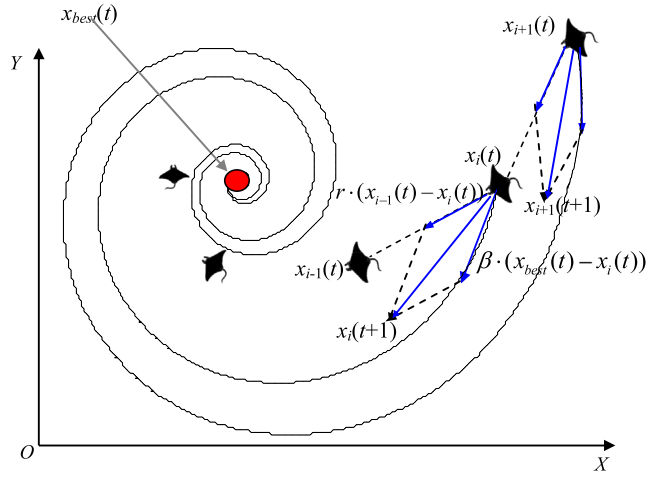

Fig. 3. Cyclone foraging behavior in a 2-D space.

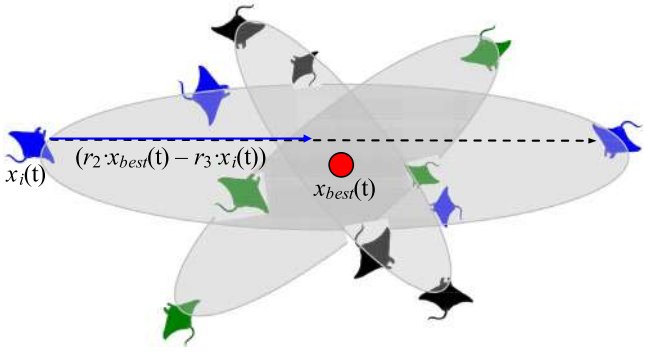

Fig. 4. Somersault foraging behavior in MRFO.

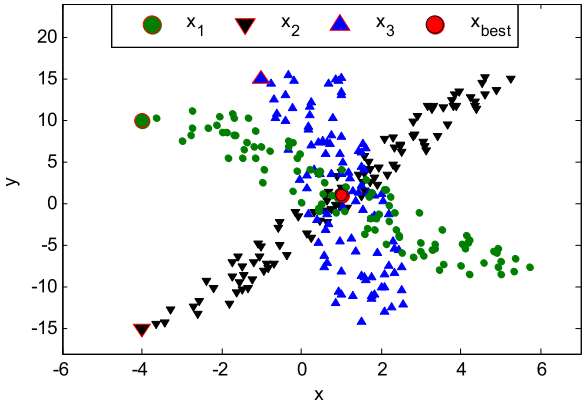

Fig. 5. Somersault foraging behavior of three individuals in a 2-D space.

2.2.3. Somersault foraging

In this behavior, the position of the food is viewed as a pivot. Each individual tends to swim to and fro around the pivot and somersault to a new position. Therefore, they always update their positions around the best position found so far. The mathematical model can be created as follows

$$x_i^d(t+1) = x_i^d(t) + S \cdot (r_2 \cdot x_{best}^d - r_3 \cdot x_i^d(t)), i = 1, \ldots, N \quad (8)$$

where S is the somersault factor that decides the somersault range of manta rays and $S = 2$, r_2 and r_3 are two random numbers in [0, 1].

As seen from Eq. (8), with definition of the somersault range, it is possible for each individual to move to any position in a new search domain located between the current position and its symmetrical position around the best position found so far. As the distance between the individual position and the best position found so far reduces, the perturbation on the current position gets reduced, too. All individuals approximate gradually to the optimal solution in the search space. Accordingly, the range of somersault foraging is adaptively reduced as iterations increase. Fig. 4 shows the sketch of somersault foraging behavior in MRFO.

Fig. 5 shows that three individuals evolved 100 times in the search space according to Eq. (8). The sampled points randomly distribute between the current positions and their symmetrical positions around x_{best}, and the sampled points become sparse as the distance reduces. The dense points around x_{best} can contribute substantially to exploitation and the sparse ones can contribute positively to exploration.

Similar to other metaheuristic optimizers, MRFO starts by generating a random population in the domain of problem. At each iteration, each individual updates its position with respect to both the one in front of it and the reference position. The value of t/T decreases from $1/T$ to 1 to respectively perform exploratory and exploitative search. The current best solution is chosen as a reference position for the exploitation when $t/T < rand$, while a random position randomly generated in the search space is chosen as a reference position for the exploration when $t/T > rand$. Meanwhile, according to the random number, MRFO can switch between the chain foraging behavior and the cyclone foraging behavior. Then individuals update their own positions with respect to the best position found so far by somersault foraging. All the update and calculation are interactively preformed until the stop criterion is met. Eventually, the position and the fitness value of the best individual are returned. The pseudocode of MRFO algorithm is given in Fig. 6.

With the description of MRFO algorithm, some remarks are noted.

(a) MRFO is motivated from three foraging strategies of manta rays, including chain foraging, cyclone foraging, and somersault foraging, which can effectively improve the optimization ability of MRFO from different aspects.

(b) The proposed chain foraging strategy makes each individual update its position with respect to the one in front of it and the current global best solution.

Initialize the size of population N, the maximal number of iterations T and each manta ray $\boldsymbol{x}_i(t)=\boldsymbol{x}_l+rand\cdot(\boldsymbol{x}_u-\boldsymbol{x}_l)$ for $i=1,\ldots,N$ and $t=1$. Compute the fitness of each individual $f_i=f(\boldsymbol{x}_i)$ and obtain the best solution found so far $\boldsymbol{x}_{best}$, Where $\boldsymbol{x}_u$ and $\boldsymbol{x}_l$ are the upper and lower boundaries of problem space, respectively .

WHILE stop criterion is not satisfied do

FOR i=1 TO N DO

IF $rand < 0.5$ THEN //*Cyclone foraging*

IF $t/T_{max} < rand$ THEN

$$\boldsymbol{x}_{rand} = \boldsymbol{x}_l + rand\cdot(\boldsymbol{x}_u - \boldsymbol{x}_l)$$

$$\boldsymbol{x}_i(t+1) = \begin{cases} \boldsymbol{x}_{rand} + r\cdot(\boldsymbol{x}_{rand} - \boldsymbol{x}_i(t)) + \beta\cdot(\boldsymbol{x}_{rand} - \boldsymbol{x}_i(t)) & i = 1 \\ \boldsymbol{x}_{rand} + r\cdot(\boldsymbol{x}_{i-1}(t) - \boldsymbol{x}_i(t)) + \beta\cdot(\boldsymbol{x}_{rand} - \boldsymbol{x}_i(t)) & i = 2,\cdots,N \end{cases}$$

ELSE

$$\boldsymbol{x}_i(t+1) = \begin{cases} \boldsymbol{x}_{best} + r\cdot(\boldsymbol{x}_{best} - \boldsymbol{x}_i(t)) + \beta\cdot(\boldsymbol{x}_{best} - \boldsymbol{x}_i(t)) & i = 1 \\ \boldsymbol{x}_{best} + r\cdot(\boldsymbol{x}_{i-1}(t) - \boldsymbol{x}_i(t)) + \beta\cdot(\boldsymbol{x}_{best} - \boldsymbol{x}_i(t)) & i = 2,\cdots,N \end{cases}$$

END IF.

ELSE //*Chain foraging*

$$\boldsymbol{x}_i(t+1) = \begin{cases} \boldsymbol{x}_i(t) + r\cdot(\boldsymbol{x}_{best} - \boldsymbol{x}_i(t)) + \alpha\cdot(\boldsymbol{x}_{best} - \boldsymbol{x}_i(t)) & i = 1 \\ \boldsymbol{x}_i(t) + r\cdot(\boldsymbol{x}_{i-1}(t) - \boldsymbol{x}_i(t)) + \alpha\cdot(\boldsymbol{x}_{best} - \boldsymbol{x}_i(t)) & i = 2,\cdots,N \end{cases}$$

END IF.

Compute the fitness of each individual $f(\boldsymbol{x}_i(t+1))$. IF $f(\boldsymbol{x}_i(t+1)) < f(\boldsymbol{x}_{best})$

THEN $\boldsymbol{x}_{best} = \boldsymbol{x}_i(t+1)$

//*Somersault foraging*

FOR i=1 TO N DO

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + S\cdot(r_2\cdot\boldsymbol{x}_{best} - r_3\cdot\boldsymbol{x}_i(t))$$

Compute the fitness of each individual $f(\boldsymbol{x}_i(t+1))$. IF $f(\boldsymbol{x}_i(t+1)) < f(\boldsymbol{x}_{best})$

THEN $\boldsymbol{x}_{best} = \boldsymbol{x}_i(t+1)$

END FOR.

END WHILE.

Return the best solution found so far $\boldsymbol{x}_{best}$.

Fig. 6. Pseudocode of MRFO algorithm.

(c) The proposed cyclone foraging strategy makes each individual update its position with respect to both the one in front of it and the reference position. Whether the best position obtained so far or a random position produced in the search space is chosen as the reference position depends on the value of t/T. The former contributes to exploitation and the latter to exploration.

(d) The gradual increase of the value of t/T encourages MRFO to smoothly transit from exploratory search to exploitative search.

(e) With the value of *rand*, MRFO can switch between the chain foraging and the cyclone foraging.

(f) The somersault foraging allows individuals to adaptively search in a changing search range.

(g) MRFO is very easy to implement and require few parameters to be adjusted.

2.3. Time complexity of the MRFO

The time complexity of MRFO depends on the number of variables, the number of individuals, and the maximum of iterations. Somersault

foraging, along with either cyclone foraging or chain foraging is performed in each iteration. Therefore, the overall time complexity of this algorithm is given as

$$O(MRFO) = O(T(O(\text{cyclone foraging} + \text{chain foraging}) + O(\text{somersault foraging}))) \quad (9)$$

$$O(MRFO) = O(T(nd + nd)) = O(Tnd) \quad (10)$$

where T is the maximum of iterations, n is the number of individuals, and d is the number of variables.

2.4. Conceptual comparison of MRFO with other optimizers

Recently, many different evolutionary algorithms inspired by the nature have been developed. Among them, there are some well-known metaheuristics which simulate the foraging behaviors and characteristics of animals including PSO, SSO, FA, and GWO.

PSO updates the velocities and positions of agents according to the following equations

$$v_i(t+1) = \omega v_i(t) + c_1 r_1(p_i - x_i(t)) + c_2 r_2(p_g - x_i(t)) \quad (11)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (12)$$

where ω is the inertia constant, c_1 and c_2 are respectively cognitive and social coefficients, p_i is the personal optimum of x_i, and p_g is the global optimum.

SSO updates the velocities and positions of sharks according to the following equations

$$Y_i^{k+1} = X_i^k + V_i^k \cdot \Delta t_k \quad i = 1, \ldots, k = 1, \ldots, k_{\max} \quad (13)$$

$$Z_i^{k+1,m} = Y_i^{k+1} + R_3 \cdot Y_i^{k+1} \quad m = 1, \ldots, M \quad (14)$$

$$V_i^k = \eta_k \cdot R_1 \frac{\delta(OF)}{\delta x_j}\bigg|_{x_{i,j}^k} + \alpha_k \cdot R_2 \cdot V_{i,j}^{k-1} \quad (15)$$

$$x_i^{k+1,m} = \text{argmax}\{OF(Y_i^{k+1}), OF(Z_i^{k+1,i}), \ldots, OF(Z_i^{k+1,M})\} \quad (16)$$

where Δt_k is interval of the time k, V_i^k is the velocity of the ith shark at time k, Y_i^{k+1} is the new position of shark due to forward movement, R_1 and R_2 are the random number in [0,1], R_3 is the random number in [−1, 1], η_k is in the interval [0,1], M is number of points in a local search, α_k is the momentum rate, and np is the size of population.

In FA, the movement of fireflies is expressed as

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha sign[rand - 1/2] \oplus Levy \quad (17)$$

$$Levy \sim u = t^{-\lambda} \quad 1 < \lambda \leq 3 \quad (18)$$

where $Levy$ is Levy distribution function, β_0 is the attractiveness at $r = 0$, α is the random parameter, and γ is an absorption coefficient. The sign $\oplus$ represents entry-wise multiplication.

In GWO, the positions of wolves that encircle the prey are updated as follows

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \quad (19)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \quad (20)$$

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1, \vec{C} = 2 \cdot \vec{r}_2 \quad (21)$$

where $\vec{X}_p$ is position vector of the food, $\vec{a}$ is linearly decreased from 2 to 0 over iterations, and r_1, r_2 are random vectors in [0, 1].

The hunting behavior of wolves can be expressed as

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_a - \vec{X}\right|, \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right| \quad (22)$$

$$\vec{X}_1 = \vec{X}_a - \vec{A}_1 \cdot \vec{D}_\alpha, \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta, \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \quad (23)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (24)$$

When modeling attacking process of wolves, GWO adaptively changes the value of $\vec{a}$ in different iterations to affect the value of $\vec{A}$. Wolves will search for prey when $|\vec{A}| \geq 1$ while wolves will attack prey when $|\vec{A}| < 1$.

Though some similarities between MRFO and these nature-inspired optimizers exist, MRFO is quite unique in many aspects. The major difference between MRFO and these optimizers is the biology inspiration. PSO is inspired by bird flocking in search of food in the sky. PSO makes use of the population share mechanism about the search space to facilitate searching for the global optimum. This makes PSO have no strong physical meaning (Bayraktar et al., 2013). SSO is originated from sharks' strong smell by which they can hunt for food based on forward movement and rotational movement. The inspiration of FA comes from the flashing communication information among fireflies, and the movement of a firefly is determined by its attractiveness between each other which is correlated with its brightness. GWO is inspired by intelligent hunting behaviors of grey wolf swarms, including encircling the prey, attacking the prey, and searching for the prey. The foraging strategies are based on social hierarchies of grey wolves. MRFO is inspired by various foraging strategies of manta rays in the ocean, including cyclone foraging, chain foraging, and somersault foraging that do not exist in other creatures. These special foraging strategies fully reflect different search behavior characteristics of manta rays, which are able to assist the optimizer in facilitating convergence to the global optimum.

The second significant difference is the mathematical equations. In PSO, in addition to the positions of particles, their velocities also need to be updated at each iteration. Moreover, each agent updates its position with respect to its velocity, the local best agent and the global best agent by adjusting the values of control parameters. This single search strategy of PSO often results in the local optima for some high-dimensional problems. In SSA, a shark updates its position according to either forward movement or rotational movement. These two movements are based on the current velocity and gradient of the considered problem. Meanwhile, there are four control parameters in the update equations of SSA and their values need to be fine-tuned. In FA, a firefly updates its position according to another firefly by adding a Levy flight. Essentially the movement of a firefly is a random step obeying power-law distribution with a heavy tail. In GWO, three prey behaviors are based on the same position update method, that is, an individual's position around the best individual is achieved with respect to the current individual by changing the values of parameters $\vec{A}$ and $\vec{C}$. In hunting behavior, the social hierarchies of grey wolves are considered including alpha, beta, and delta levels. In MRFO, for the chain foraging and cyclone foraging, the individual is updated based on the former individual and the food by introducing two weight coefficients, by which the chain motion and spiral motion are simulated. In somersault foraging, the individual is updated based on the food by using two random numbers. Additionally, these diverse search strategies contribute significantly to search for the global optimum extensively and intensively. To the best of our knowledge, these significant differences in the methods and equations between MRFO and the other optimizers show that none of meta-heuristics in the literature can mimic the foraging behaviors of manta rays.

3. Experimental results and discussion

3.1. Benchmark functions and experimental setup

MRFO algorithm is tested with 31 well-known benchmark functions. The first 23 functions are classical benchmark functions used in literature (Zhao and Wang, 2016; Digalakis and Margaritis, 2011; Liang et al., 2013). These benchmark functions have been often employed to compare the performance of metaheuristics. These functions are summarized in Tables A.1–A.3. For Table A.1 are unimodal functions each of which has not local optima. Tables A.2 and A.3 are multimodal

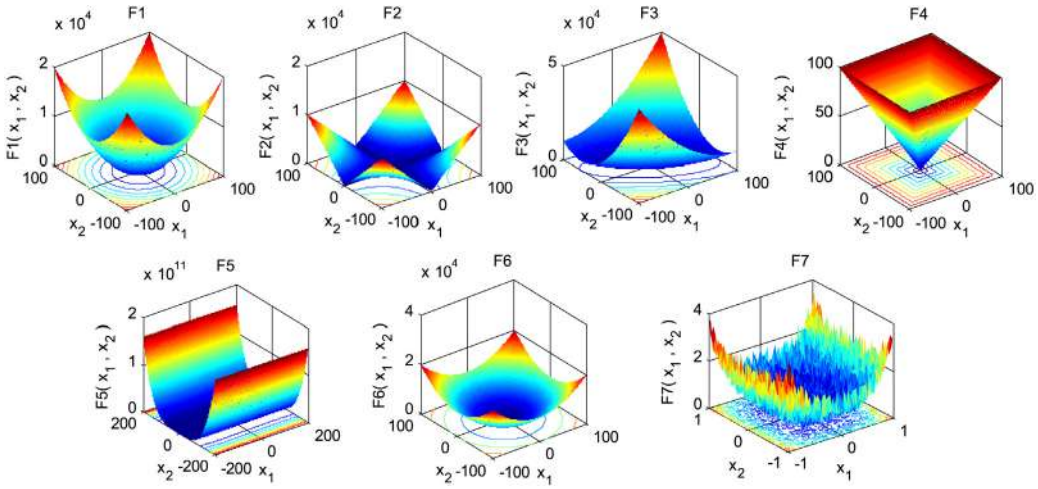

Fig. 7. 3-D maps for 2-D unimodal benchmark functions.

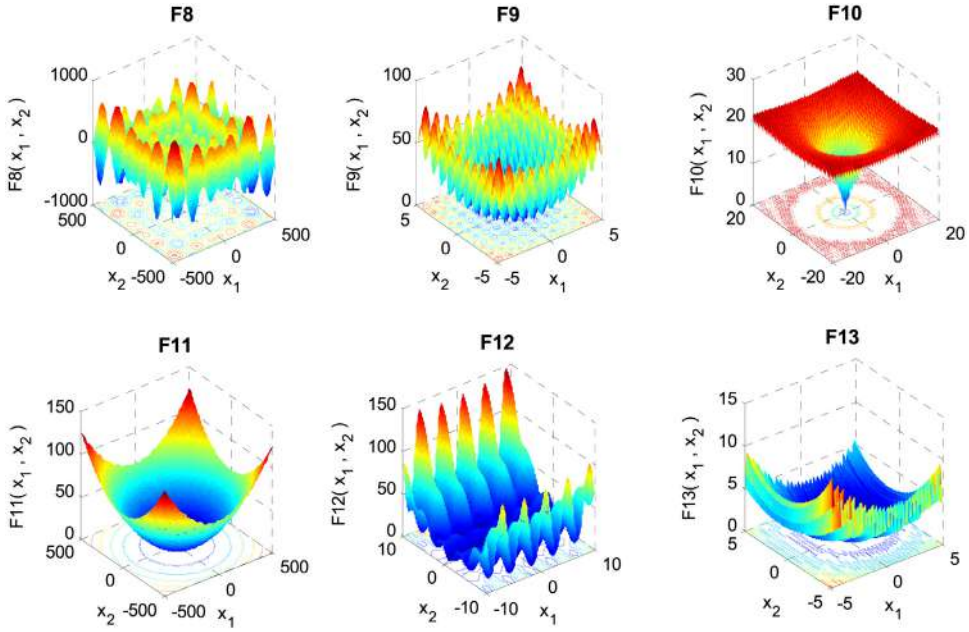

Fig. 8. 3-D maps for 2-D multimodal benchmark functions.

and low-dimensional functions that have more than one local optima. Table A.4 are eight composition benchmark functions compiled in the CEC 2014 special session (Liang et al., 2013). Because each composition function is composed of basic and hybrid functions, they are well suited to testing the potential performance of algorithms. The 3-D maps for these different types of 2-D benchmark functions are shown in Figs. 7–10, respectively.

We compare MRFO method with six well-established optimizers, including GA, PSO, DE, CS, ABC, and GSA. The initial parameter settings of these optimizers are provided in Table 1. For all considered algorithms, the population size and the maximum number of function evaluations (FEs) are selected as 30 and 50,000, respectively. Additionally, every algorithm performs 30 runs for each function and the results are based on the average performance of these runs.

Table 1
Initial parameter settings for all algorithms.

Algorithm	Parameter	Value
CS	Mutation probability	0.25
DE	Mutation factor	0.5
	Crossover rate	0.5
PSO	Cognitive coefficient	2
	Social coefficient	2
	Inertia constant	Linearly decreases from 0.8 to 0.2
GSA	Gravitational constant	100
	Decreasing coefficient	20
GA	Selection	Roulette wheel
	Crossover rate	0.8
	Mutation rate	0.2
ABC	Limit	$SN \cdot D$

3.2. Analysis of exploitation performance

Fig. 11 provides the bar charts obtained from MRFO and other algorithms on unimodal functions and their convergence curves are shown in Fig. 12. According to Fig. 11, MRFO shows its strong competitiveness in exploitation, outperforming the other metaheuristic algorithms for each unimodal function. Although Function f_5 is a non-convex

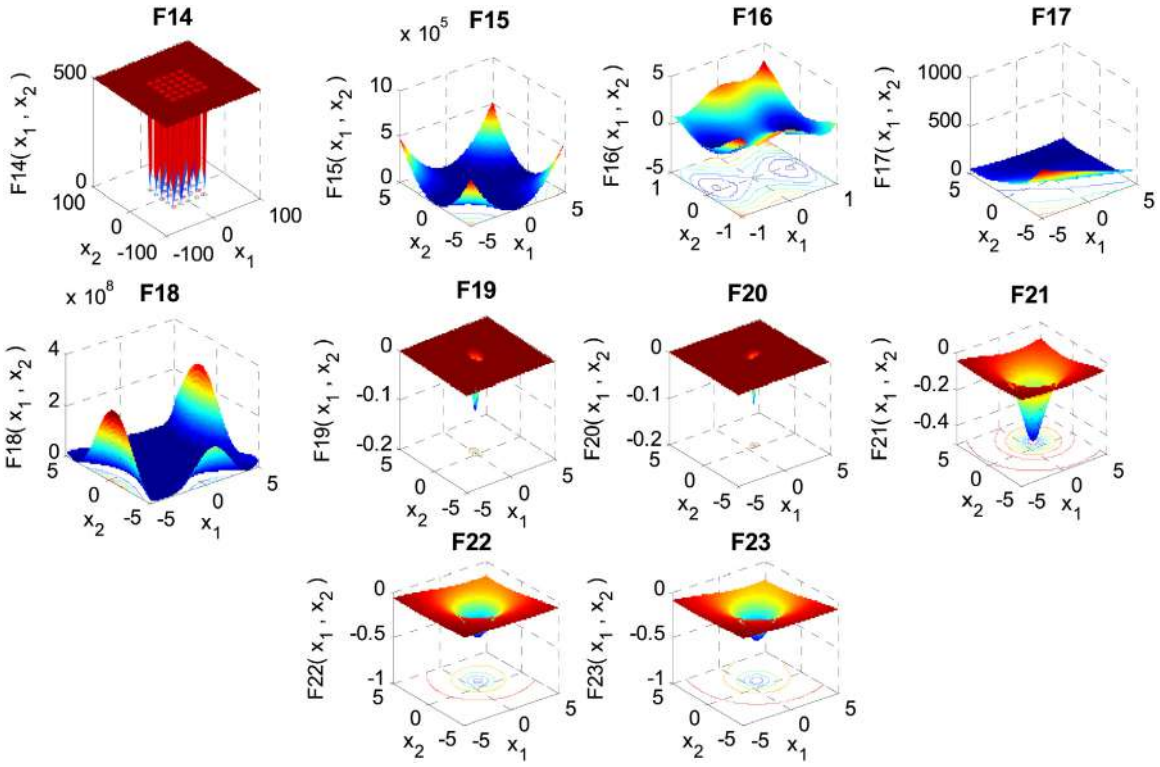

Fig. 9. 3-D maps for 2-D low-dimensional multimodal benchmark functions.

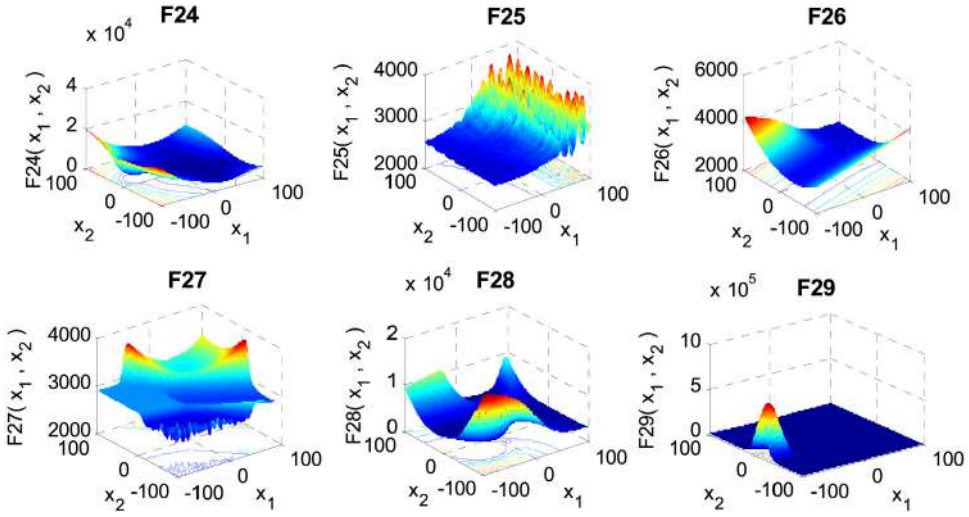

Fig. 10. 3-D maps for 2-D composition benchmark functions.

quadratic function, MRFO still performs the best in terms of exploitation and convergence ability in Fig. 12. For exploitation, CS performs the worst for functions f_1 and f_2, ABC performs the worst for functions f_3, f_4 and f_5, and, PSO and DE perform the worst for functions f_6 and f_7. PSO has the slowest convergence rate for functions f_3, f_4, and f_5, and CS converges the slowest for function f_2. The results suggest that MRFO is the most efficient method in tackling unimodal functions.

3.3. Analysis of exploration performance

Fig. 13 gives the bar charts obtained from MRFO and other algorithms on multimodal functions and their convergence curves are depicted in Fig. 14. As shown in Fig. 13, MRFO performs better than other heuristic optimizers for all the multimodal functions except function f_{13}. Although MRFO cannot perform the best for function f_{13}, it outperforms ABC, demonstrating its superior exploration. Though function f_{11} has many widespread and regularly distributed local minima, MRFO is extremely successful in stepping out of them and the final results are satisfactory. GSA provides the worst results for functions f_8 and f_{11}, ABC performs the worst for functions f_{12} and f_{13}, and, DE and CS find the worst solutions for functions f_9 and f_{10}, respectively. Similarly, as shown in Fig. 14, MRFO has a better convergence than the other optimizers for all the multimodal functions except function f_{13}.

The bar charts obtained from MRFO and other algorithms on low-dimensional multimodal functions and their convergence curves are depicted in Figs. 15 and 16. According these results, all the algorithms perform similarly for functions f_{16}, f_{17}, f_{18}, and f_{19}, they also similarly perform well for function f_{14} except for GA and GSA. Additionally, MRFO is only inferior to CS and DE on function f_{15}, while it is

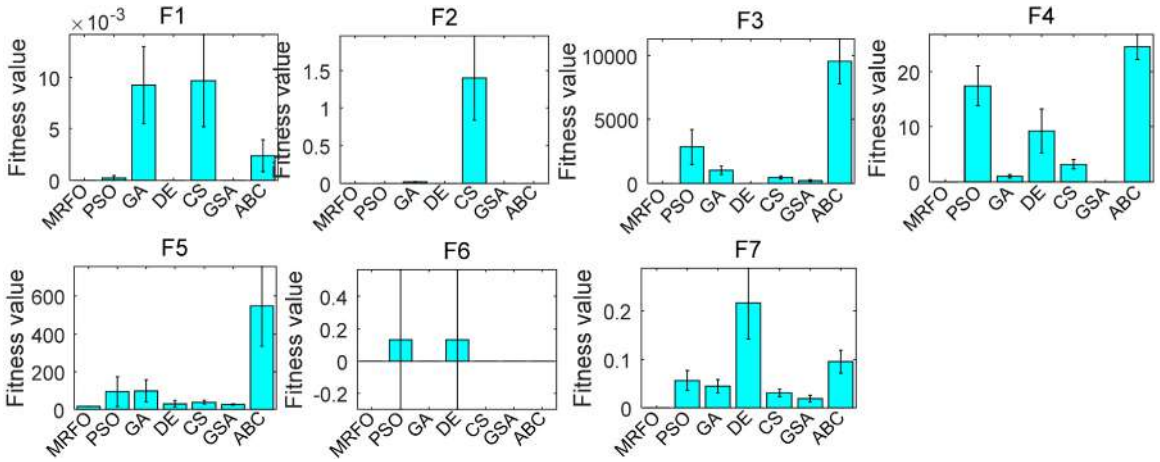

Fig. 11. Bar charts obtained from MRFO and other algorithms on unimodal functions.

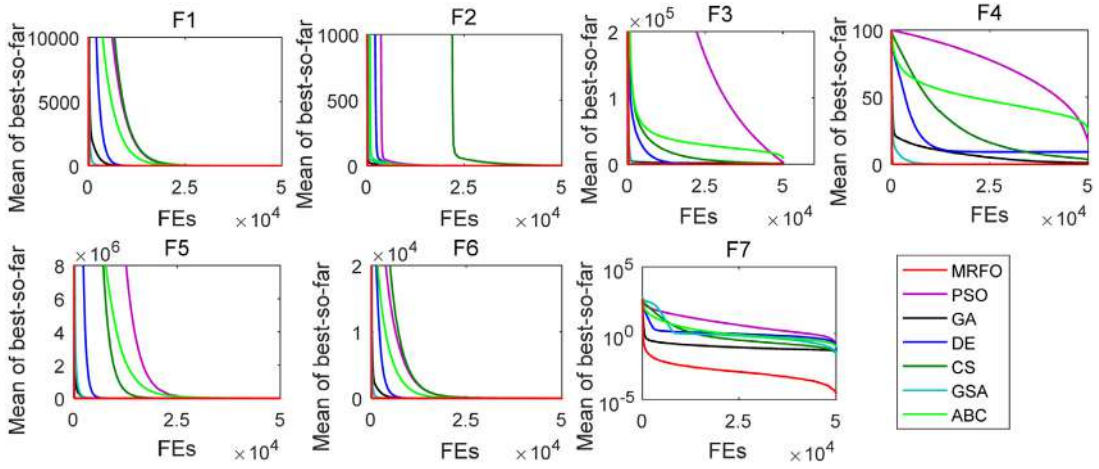

Fig. 12. Convergence comparisons of algorithms for unimodal functions.

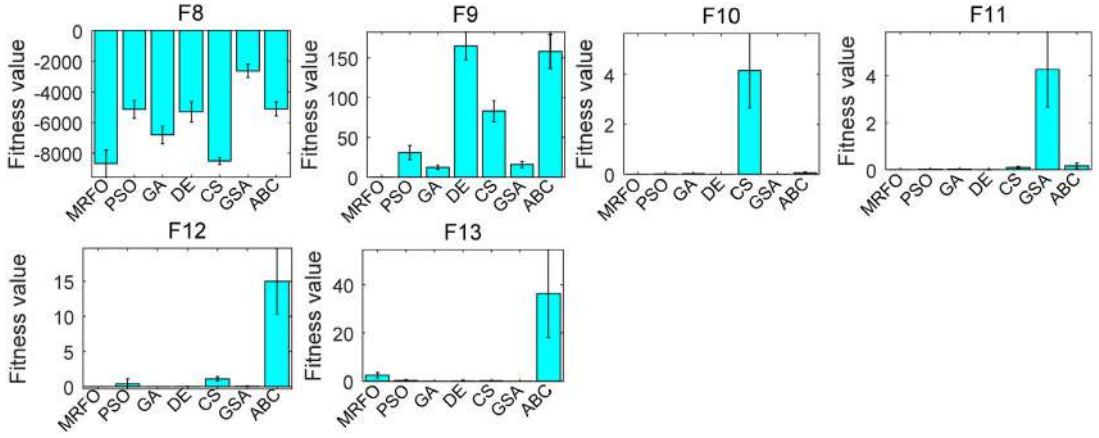

Fig. 13. Bar charts obtained from MRFO and other algorithms on multimodal functions.

superior to PSO on functions f_{20}, f_{21}, and f_{23}. MRFO outperforms its competitors on the majority of the low-dimensional functions.

3.4. Analysis of local optima avoidance

The composite problems are especially suited to evaluating the balance between exploration and exploitation. The bar charts obtained from MRFO and other algorithms on composite functions and their convergence curves are depicted in Figs. 17 and 18. From Fig. 17, MRFO performs the best for functions f_{24}, f_{25}, f_{26}, f_{28}, and f_{29}, it outperforms GA and GSA for functions f_{27}. In addition, it is superior to GA and PSO for function f_{30}. Moreover, MRFO is very competitive with other optimizers in terms of convergence ability according to Fig. 18. All the algorithms except MRFO are trapped into local optima at different levels for functions f_{24}, f_{25}, f_{26}, f_{28}, and f_{29}. The cyclone foraging of

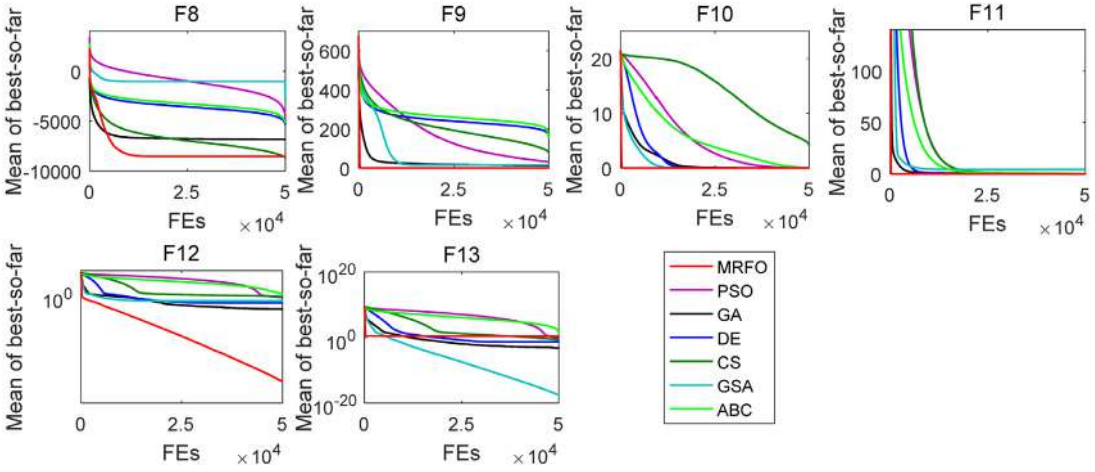

Fig. 14. Convergence comparisons of algorithms for multimodal functions.

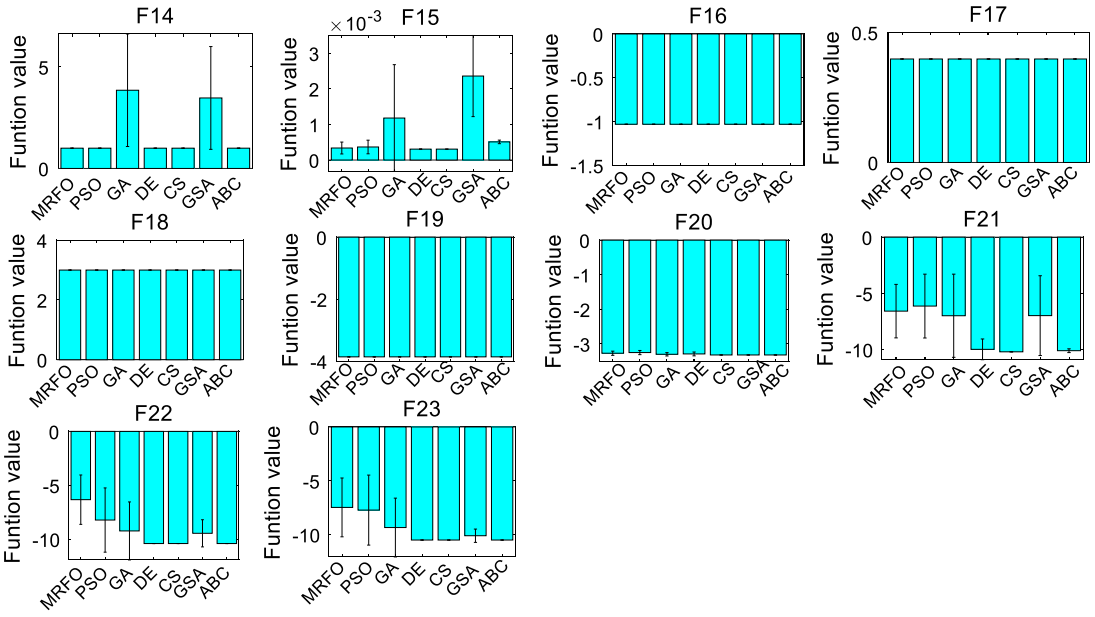

Fig. 15. Bar charts obtained from MRFO and other algorithms on low-dimensional functions.

MRFO may contribute to exploration and efficiently avoid local optima. The results demonstrate that MRFO is effective in balancing exploratory and exploitative search and guaranteeing the global convergence.

3.5. Analysis of statistical significance

Wilcoxon Signed-Rank Test (WSRT), as a nonparametric test, can effectively assess statistical significance difference between two optimizers. The statistical results of WSRT on 31 benchmark functions in 30 runs are presented in Tables 2 and 3, where T+ and T− are calculated and their *p*-values can be obtained. '=' indicates the case in which there is no significance difference between MRFO and its competitor, '+' shows that MRFO possesses better performance than the comparative algorithm at 95% significance level ($a = 0.05$) and '−' vice versa. The corresponding statistical results of '+', '−' and '=' for each function in 30 runs are listed in Table 4. As shown in the table, the results highlight the obvious superiority of MRFO to all the other competitors on four different types of benchmark functions.

3.6. Performance measures

To fully evaluate optimization performance of the proposed algorithm, two performance measures, including average solution cost (ASC) and success ratio (SR), are employed in this experiment. ASC represents the average costs of an algorithm when finding the global optimum. SR is the probability of an algorithm that can find the global optimum. These two measures are defined as (Liu et al., 2005)

$$ASC = \frac{\sum_{I=1}^{N_r} m_i}{N_r} \tag{25}$$

$$SR = \frac{N_r}{50} \times 100\% \tag{26}$$

where N_r is the number of success runs during N independent runs and m_i is the number of function evaluations of the ith success run. In MRFO algorithm, the maximum evaluation number of functions is set as 50,000 and the algorithm is run 50 times independently. If the algorithm reaches the global optimum within a gap of 0.001 as used in Karaboga and Akay (2009), its evaluation number in this search run is recorded.

The ASC and SR of all the algorithms are provided in Table 5 when solving the 23 benchmark functions. From this table, ABC costs

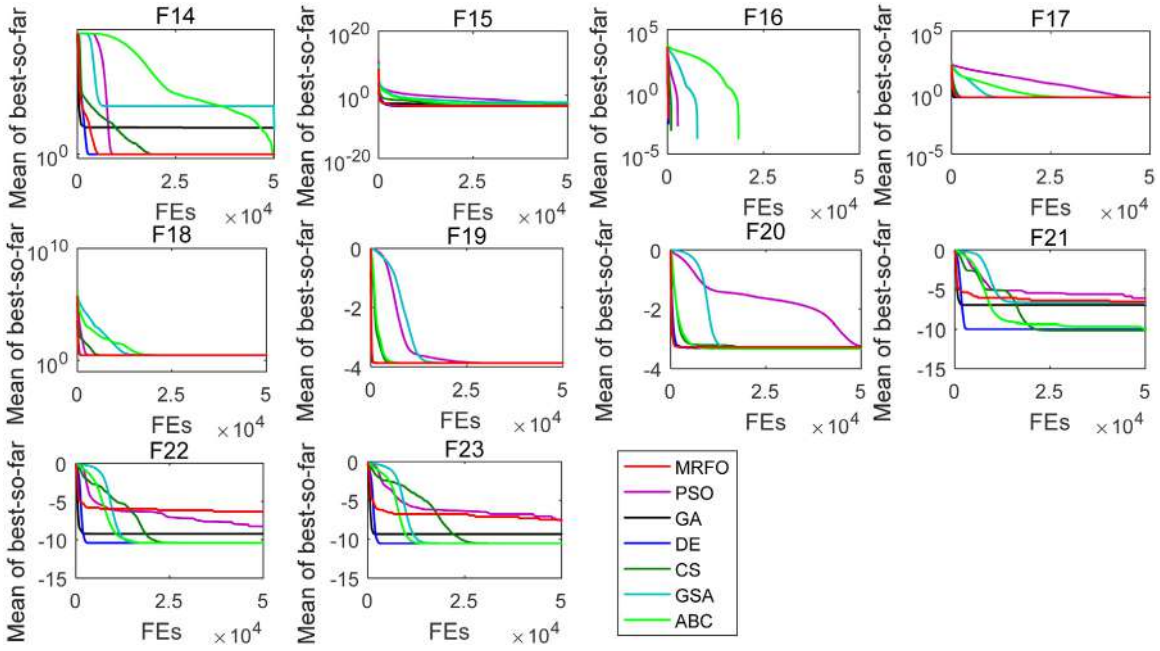

Fig. 16. Convergence comparisons of algorithms for low-dimensional functions.

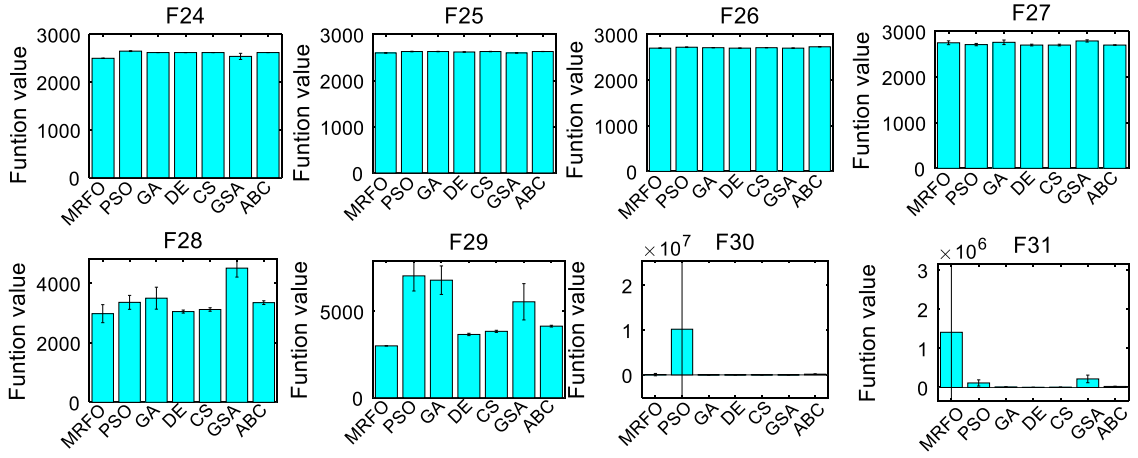

Fig. 17. Bar charts obtained from MRFO and other algorithms on composition functions.

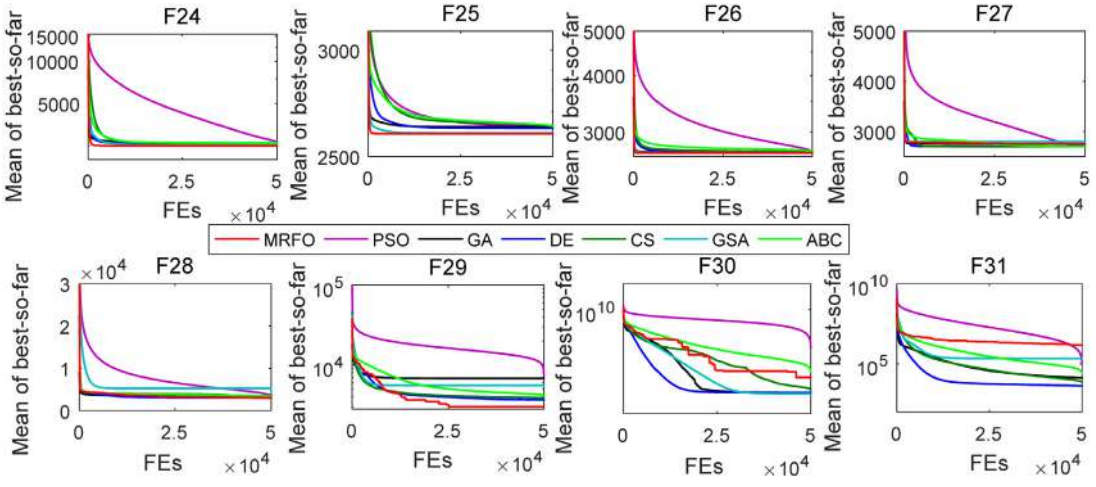

Fig. 18. Convergence comparisons of algorithms for composition functions.

Table 2
Statistical comparisons of WSRT for MRFO vs. GA, PSO, and DE.

Function	MRFO vs. PSO				MRFO vs. GA				MRFO vs. DE			
	p-value	T−	T+	Winner	*p*-value	T−	T+	Winner	p-value	T−	T+	Winner
$f_1(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_2(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_3(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_4(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_5(x)$	1.92E−06	1	464	−	2.35E−06	3	462	−	2.84E−05	29	436	−
$f_6(x)$	1	0	465	=	2.50E−01	0	465	=	2.50E−01	0	465	=
$f_7(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_8(x)$	3.52E−06	7	458	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_9(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{10}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{11}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{12}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{13}(x)$	1.92E−06	464	1	+	4.73E−06	455	10	+	1.92E−06	464	1	+
$f_{14}(x)$	1.22E−05	0	465	−	1.00E+00	0	465	0	1	0	465	=
$f_{15}(x)$	1.02E−05	18	447	−	2.84E−05	29	436	−	2.34E−06	435	30	+
$f_{16}(x)$	1.25E−01	0	465	=	1.00E+00	0	465	=	1	0	465	=
$f_{17}(x)$	3.91E−03	0	465	−	1.00E+00	0	465	=	11	0	465	=
$f_{18}(x)$	4.11E−04	2	463	−	7.50E−01	6.5	458.5	=	3.13E−02	33	432	−
$f_{19}(x)$	3.79E−06	0	465	−	1.00E+00	0	465	=	1	0	465	=
$f_{20}(x)$	9.59E−01	235	230	=	1.77E−02	99	366	−	1	95	370	=
$f_{21}(x)$	7.81E−01	246	219	=	7.05E−01	235	230	=	3.76E−05	231	234	−
$f_{22}(x)$	1.48E−03	387	78	+	5.19E−03	346	119	+	1.02E−05	276	189	+
$f_{23}(x)$	1.65E−01	300	165	=	6.26E−01	240	225	=	6.98E−05	190	275	−
$f_{24}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{25}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{26}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{27}(x)$	8.77E−01	240	225	=	1.13E−05	446	19	+	1.92E−06	464	1	+
$f_{28}(x)$	1.60E−04	49	416	−	1.80E−05	24	441	−	3.59E−04	59	406	−
$f_{29}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{30}(x)$	6.73E−01	212	253	=	1.73E−06	0	465	−	6.73E−01	212	253	=
$f_{31}(x)$	5.67E−03	367	98	+	2.85E−02	339	126	+	3.38E−03	375	90	+

Table 3
Statistical comparisons of WSRT for MRFO vs. CS, GSA, and ABC.

Function	MRFO vs. CS				MRFO vs. GSA				MRFO vs. ABC			
	p-value	T−	T+	Winner	*p*-value	T−	T+	Winner	p-value	T−	T+	Winner
$f_1(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_2(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_3(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_4(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_5(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_6(x)$	1	0	465	=	1	0	465	=	1	0	465	=
$f_7(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_8(x)$	0.349346	278	187	=	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_9(x)$	1.73E−06	0	465	−	1.72E−06	0	465	−	1.73E−06	0	465	−
$f_{10}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{11}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{12}(x)$	1.73E−06	0	465	−	1.72E−06	0	465	−	1.73E−06	0	465	−
$f_{13}(x)$	5.75E−06	453	12	+	8.89E−07	464	1	+	1.73E−06	0	465	−
$f_{14}(x)$	1	0	465	=	2.56E−06	0	465	−	1.73E−06	0	465	−
$f_{15}(x)$	3.11E−05	30	435	−	1.73E−06	0	465	−	3.11E−05	30	435	−
$f_{16}(x)$	1	0	465	=	2.21E−05	0	465	−	5.00E−01	0	465	=
$f_{17}(x)$	1	0	465	=	1	0	465	=	1.73E−06	0	465	−
$f_{18}(x)$	0.007324	71	394	−	6.78E−06	0	465	−	1.73E−06	0	465	−
$f_{19}(x)$	1	0	465	=	2.21E−05	0	465	−	5.99E−07	0	465	−
$f_{20}(x)$	0.04715	329	136	+	0.003416	231	234	−	8.54E−03	259	206	+
$f_{21}(x)$	0.000108	420	45	+	0.688262	236	229	=	1.15E−04	420	45	+
$f_{22}(x)$	2.43E−05	437	28	+	1.38E−05	437	28	+	2.58E−05	437	28	+
$f_{23}(x)$	0.003595	374	91	+	0.000445	311	154	+	3.61E−03	374	91	+
$f_{24}(x)$	1.73E−06	0	465	−	0.115608	156	309	=	1.73E−06	0	465	−
$f_{25}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{26}(x)$	1.73E−06	0	465	−	7.52E−02	146	319	=	1.73E−06	0	465	−
$f_{27}(x)$	1.73E−06	465	0	+	1.89E−04	51	414	−	1.73E−06	465	0	+
$f_{28}(x)$	3.88E−04	60	405	−	1.92E−06	1	464	−	3.59E−04	59	406	−
$f_{29}(x)$	1.73E−06	0	465	−	1.73E−06	0	465	−	1.73E−06	0	465	−
$f_{30}(x)$	0.071903	145	320	=	2.43E−02	342	123	+	3.11E−05	30	435	−
$f_{31}(x)$	3.61E−03	374	91	+	4.72E−02	329	136	+	8.73E−03	360	105	+

less on 3 functions, PSO costs less on 2 functions, GA costs less on 1 function, DE costs less on 2 functions, CS costs less on 1 functions and GSA costs less on 1 function, while MRFO costs less on 11 functions. Obviously, MRFO is able to cost the least average function evaluation number when finding the global solution with given accuracy. For the SC, MRFO provides the highest success rate with 100% on 15 functions; ABC provides the highest success rate with 100% on 13 functions; PSO provides the highest success rate on 9 functions; GA provides the

Table 4
Statistical results of WSRT of MRFO.

Function type	MRFO vs. GA (+/=/−)	MRFO vs. PSO (+/=/−)	MRFO vs. DE (+/=/−)	MRFO vs. CS (+/=/−)	MRFO vs. GSA (+/=/−)	MRFO vs. ABC (+/=/−)
Unimodal	6/1/0	6/1/0	6/1/0	6/1/0	6/1/0	6/1/0
Multimodal	5/0/1	5/0/1	5/0/1	4/1/1	5/0/1	6/0/0
Low-dimension	5/4/1	2/7/1	3/5/2	2/4/4	6/2/2	5/1/4
Composition	5/2/1	6/0/2	5/1/2	5/1/2	4/2/2	6/0/2
Total	21/7/3	19/8/4	19/7/5	17/7/7	21/5/5	23/2/6

Table 5
Statistical results of ASC and SR from different algorithms.

No	ABC		PSO		GA		DE		CS		GSA		MRFO	
	ASC	SR	ASC	SR	ASC	SR	ASC	SR	ASC	SR	ASC	SR	ASC	SR
f_1	48,553	100	40,674	100	45,093	12	15,716	100	45,115	100	16,552	100	498	100
f_2	42,091	100	42,812	100	48,721	4	20,901	100	–	0	24,823	100	503	100
f_3	–	0	–	0	–	0	–	0	–	0	–	0	461	100
f_4	–	0	–	0	–	0	–	0	–	0	21,152	100	628	100
f_5	–	0	–	0	–	0	–	0	–	0	–	0	–	0
f_6	35,452	100	15,951	66	–	0	15,812	100	41,153	100	5,016	100	379	100
f_7	–	0	–	0	38,901	16	–	0	48,122	32	–	0	821	100
f_8	–	0	–	0	–	0	–	0	–	0	–	0	–	0
f_9	–	0	–	0	–	0	–	0	–	0	–	0	781	100
f_{10}	–	0	16,912	90	26,711	32	20,966	100	–	0	22,552	100	519	100
f_{11}	–	0	13,812	92	32,551	26	38,961	38	–	0	–	0	–	0
f_{12}	–	0	42,568	32	16,189	100	26,436	82	–	0	18,667	78	3,026	100
f_{13}	–	0	41,901	16	36,966	100	40,164	36	–	0	13,172	100	46,589	6
f_{14}	11,631	100	1,006	100	47,171	6	20,913	100	5,800	100	48,413	2	1,081	100
f_{15}	1,192	100	1,206	100	1,291	100	1,328	100	1,362	100	33,271	56	690	100
f_{16}	1,373	100	1,891	100	981	100	586	100	1,421	100	5,601	100	591	100
f_{17}	1,421	100	1,377	100	1,069	100	877	100	1,381	100	5,192	100	898	100
f_{18}	18,731	100	1,171	100	1,519	100	1,071	100	1,816	100	14,812	100	901	100
f_{19}	1,558	100	1,526	100	580	100	882	100	2,191	100	15,211	100	679	100
f_{20}	1,290	100	16,925	52	33,115	82	35,597	74	15,751	100	16,337	100	28,162	68
f_{21}	6,428	100	26,312	56	32,160	72	21,413	82	15,911	100	44,705	24	24,162	58
f_{22}	7,656	100	26,919	58	34,801	64	18,111	100	16,161	100	17,311	92	27,805	54
f_{23}	18,041	100	23,892	62	36,905	68	26,915	100	14,878	100	15,003	98	28,280	46
Mean	15,032	57	18,639	58	25,572	47	18,038	66	16,236	54	19,870	63	8,373	75

highest success rate with 100% on 7 functions; DE provides the highest success rate with 100% on 12 functions; CS provides the highest success rate with 100% on 12 functions; GSA provides the highest success rate with 100% on 11 functions. These results suggest that MRFO algorithm may locate the global optimum with very high probability for most benchmark functions. Especially, MRFO can achieve the predetermined solution accuracy with very small computational expense for unimodal functions. Therefore, the statistical results of these performance measures on the test functions demonstrate the effectiveness and reliability of MRFO algorithm.

In summary, the above results discover different characteristics of MRFO in dealing with different types of benchmark functions. In earlier iterations, as shown in Eq. (7), the cyclone foraging strategy based on random positions requires each individual to update its position according to a randomly generated position in the entire search space, enjoying an extensive exploration. As iterations pass, however, an intensive exploitation and a good convergence are gradually highlighted which are implied in Eq. (1), this chain foraging strategy forces individuals to rapidly move to the best position obtained so far. Each of these two strategies is separately performed and spends almost half of iterations. Meanwhile, to strengthen exploration and exploitation as well as convergence of MRFO in each iteration, the somersault foraging strategy, as Eq. (8) reveals, can allow individuals to adaptively perform either a local search around the best position obtained so far or a global search between the current position and the best position found so far. Obviously, the results testing the performance of MRFO compared with its competitors in handling a range of different functions show that, the proposed algorithm can successfully and automatically balance exploration and exploitation, and its convergence performance is improved simultaneously in iterations.

4. MRFO for real engineering problems

In order to further evaluate MRFO, eight classical engineering problems from literature are solved using MRFO. The other algorithms are also used to address these real engineering problems. A comprehensive comparison of MRFO and the others algorithms demonstrates the practicability and superiority of MRFO in handling challenging engineering problems. The eight constrained engineering design problems are employed to evaluate MRFO.

Generally, these constrained engineering optimization problems (in minimization case) can be formulated as

$$\text{Minimize } f(\vec{x}), \quad \vec{x} \in R^d \tag{27}$$

$$\text{Subject to } \begin{cases} g_i(\vec{x}) \le 0 & i = 1, \dots, p \\ h_j(\vec{x}) = 0 & j = 1, \dots, q \end{cases} \tag{28}$$

where g_i and h_i are the inequality and equality constraints, respectively, and R^d is the n-dimensional field of real numbers. The task of MRFO is to find the best feasible solution $\vec{x} = \{x_1, \dots, x_d\}$ which minimizes the objective function $f(\vec{x})$ with constraints.

In this study, the penalty function is employed to deal with these constraints in MRFO. A considerately large penalty value proportional to the values of the violated constraints is added to the objective function. Therefore, the optimization of these engineering problems using MRFO in Eq. (27) is defined as

$$\text{Minimize } F(\vec{x}) = \begin{cases} f(\vec{x}) & \vec{x} \in S \\ f(\vec{x}) + \lambda\left(\sum_{i=1}^{p} g_i(\vec{x}) + \sum_{j=1}^{q} h_j(\vec{x})\right) & \vec{x} \notin S \end{cases} \tag{29}$$

where S is the feasible search space. When using this method, individuals who violate any constraint with any level are assigned a big

Table 6
Result comparisons of seven optimizers for tension/compression spring design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
$x_1(d)$	0.0523734	0.0578383	0.0589611	0.0545403	0.0560332	0.0522907	0.0528372
$x_2(D)$	0.3733461	0.4979163	0.5581978	0.4292681	0.4700379	0.3644606	0.3733680
$x_3(N)$	10.3831265	6.7114619	4.9883445	8.0300516	6.8152103	11.7484375	10.7654571
g_1	−0.0004310	−0.0313137	−0.0000601	−0.0000025	−0.0001470	−0.0597448	−0.0015036
g_2	−0.0001276	−0.0415641	−0.0000282	0.0000000	−0.0008437	−0.0150534	−0.0245475
g_3	−4.0825408	−3.8821081	−4.3278787	−4.1768253	−4.2266307	−3.7061375	−3.9448645
g_4	−0.7161870	−0.6294970	−0.5885608	−0.6774611	−0.6492860	−0.7221658	−0.7158632
f_1	0.0126813	0.0145104	0.0135610	0.0128076	0.0130094	0.0137010	0.0133061

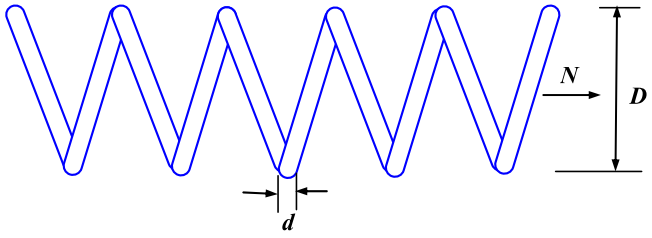

Fig. 19. Tension/compression string design problem.

function fitness value. Thus, the infeasible solutions will be automatically discarded by the algorithm during the optimization process. In this way, a constrained problem is converted into an unconstrained problem by introducing a penalty function. This method is very effective and is easy to implement for MRFO without the need for modification of the algorithm.

In this subsection, the results are compared with those of the optimizers used previously on these engineering problems, to be fair, all the optimizers adopt the same FEs for every engineering problem and are equipped with the same penalty function. Second, the results are also compared with those of the other widely used approaches in literature, in which different constraint handling strategies are employed. The optimization formulation of these engineering problems is given in Appendix B.

4.1. Tension/compression spring design

This problem (Belegundu, 1982), depicted in Fig. 19, requires the minimization of the weight of a tension/compression spring with three constraints. There are three variables, including wire diameter (d), mean coil diameter (D), and number of active coils (N), to be optimized.

This case is solved using MRFO and the ones mentioned above with 40,000 FEs. The results in terms of decision variables, constraint values and function values are summarized in Table 6, where MRFO offers better results than the other approaches. Meanwhile, Table 7 lists the comparisons for this problem using MRFO and those optimization approaches in literature, such as GA2 (Coello, 2000a,b), GA3 (Coello and Montes, 2002), CA (Coello and Becerra, 2004), PSO2 (He and Wang, 2007), CPSO (He and Wang, 2006), HPSO (He and Wang, 2007), QPSO (Dos Santos Coelho, 2010), UPSO (Parsopoulos and Vrahatis, 2005), CDE (Huang et al., 2007), SSB (Ray and Liew, 2003), and $(\mu + \lambda)$ES (Mezura-Montes and Coello, 2005). In Table 7, the two sets of results are provided using MRFO for 2000 FEs and 50,000 FEs, respectively. MRFO method under pre-defined maximum function evaluations is significantly competitive. Moreover, MRFO has obtained the best results in terms of mean solutions using less number of FEs compared with the other considered competitors. The function and constraint values versus FEs are illustrated Fig. 20, respectively.

This problem, proposed by Kannan and Kramer (1994), requires the minimization of the fabrication of a cylindrical pressure vessel and meets four constraints. The decision variables depicted in Fig. 21 consists of thickness of weld (T_s), length of the clamped bar (T_h), height of the bar (R), and thickness of the bar (L).

Table 7
Result comparisons of optimizers in literature. "NA" stands for not available.

Methods	Worst	Mean	Best	Std	FEs
GA2	0.0128221	0.0127690	0.0127047	3.9390E−05	900,000
GA3	0.0129730	0.0127420	0.0126810	5.9000E−05	80,000
CA	0.0151156	0.0135681	0.0127210	8.4215E−04	50,000
CPSO	0.0129240	0.0127300	0.0126747	5.1985E−04	200,000
HPSO	0.0127191	0.0127072	0.0126652	1.5824E−05	75,000
PSO2	0.0718020	0.0195550	0.0128570	0.0116620	2 000
QPSO	0.0181270	0.0138540	0.0126690	1.3410E−03	2 000
UPSO	0.0503651	0.0229478	0.0131200	7.2057E−03	10,000
CDE	0.0127900	0.0127030	0.0126702	2.7000E−05	240,000
SSB	0.0167173	0.0129227	0.0126692	5.9000E−04	25,167
$(\mu + \lambda)$ES	NA	0.0131650	0.0126890	3.9000E−04	30,000
MRFO	0.0142764	0.0134659	0.0129729	2.7718E−03	2 000
MRFO	0.0131811	0.0127007	0.0126757	2.1378E−04	50,000

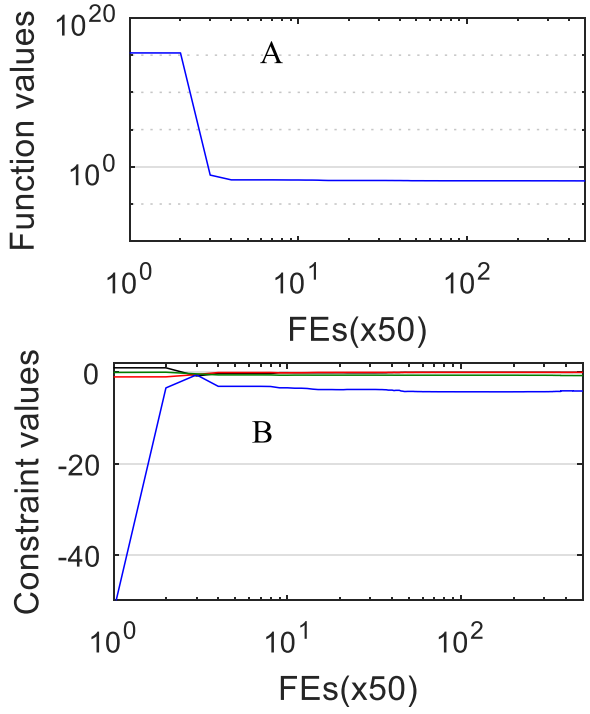

Fig. 20. (A) Function values versus FEs, and (B) constraint values versus FEs.

4.2. Pressure vessel design

Table 8 offers the results of seven algorithms with 50,000 FEs in terms of decision variables, constraint values and function values. Observing the table, MRFO is the second most efficient approach only after CS. This problem is also solved using other well-established methods, such as GA2 (Coello, 2000a,b), GA3 (Coello and Montes, 2002), CPSO (He and Wang, 2006), HPSO (He and Wang, 2007), PSO-DE (Liu et al., 2010), PSO2 (He and Wang, 2007), CDE (Huang et al., 2007), QPSO (Dos Santos Coelho, 2010), ABC (Akay and Karaboga, 2012a,b),

Table 8
Result comparisons of seven optimizers for pressure vessel design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
$x_1(T_s)$	0.7786521	1.5229311	0.8947263	0.8772875	0.7781832	1.1066323	0.9845336
$x_2(T_h)$	0.3848881	0.9353978	0.4422071	0.4336437	0.3846669	8.9285040	0.4832150
$x_3(R)$	40.3446679	69.2898207	46.3500159	45.4553115	40.3203716	57.3384588	49.6633562
$x_4(L)$	199.6515915	55.2699885	130.2387310	139.0505211	199.9895907	77.4829446	102.7808834
g_1	0.0000000	−0.1856375	−0.0001710	−3.61E−10	−4.25E−09	−2.88E−10	−0.0260309
g_2	0.0000000	−0.2743729	−0.0000280	−1.13E−09	−1.05E−05	−8.3814952	−0.0094266
g_3	−0.0000034	−9.31E+05	−100.0362879	−3.8444E−03	−0.3607189	−2.9393E+05	−1.3500E+04
g_4	−40.3484085	−184.7300115	−109.7612690	−100.9494789	−40.0104093	−162.5170554	−137.2191166
f_2	5.8862E+03	1.5210E+04	6.1171E+03	6.0773E+03	5.8854E+03	5.6948E+04	6.5176E+03

Table 9
Result comparisons of optimizers in literature.

Methods	Worst	Mean	Best	Std	FEs
GA2	6308.4970	6293.8432	6288.7445	7.4133	900,000
GA3	6469.3220	6177.2533	6059.9463	130.9297	80,000
CPSO	6363.8041	6147.1332	6061.0777	86.4500	240,000
HPSO	6288.6770	6099.9323	6059.7143	86.2000	81,000
PSO-DE	NA	6059.7140	6059.714	NA	42,100
PSO2	14 076.3240	8756.6803	6693.7212	1492.5670	8 000
QPSO	8017.2816	6440.3786	6059.7209	479.2671	8 000
CDE	6371.0455	6085.2303	6059.7340	43.0130	204,800
ABC	NA	6245.3080	6059.7140	205.0000	30,000
$(\mu + \lambda)$ES	NA	6379.9380	6059.7010	210.0000	30,000
CSA	7332.8410	6342.4990	6059.7140	384.9450	250,000
MRFO	6366.6491	6167.490	5987.8131	12.6209	8 000
MRFO	6152.2926	6046.3101	5889.1755	2.5318	30,000

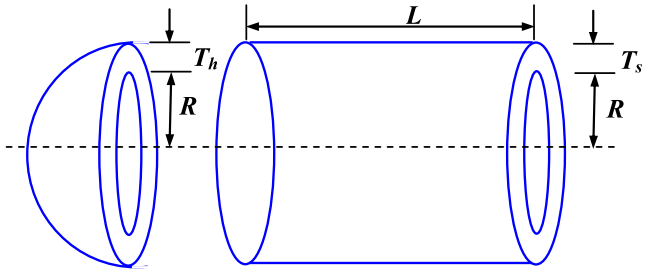

Fig. 21. Pressure vessel design problem.

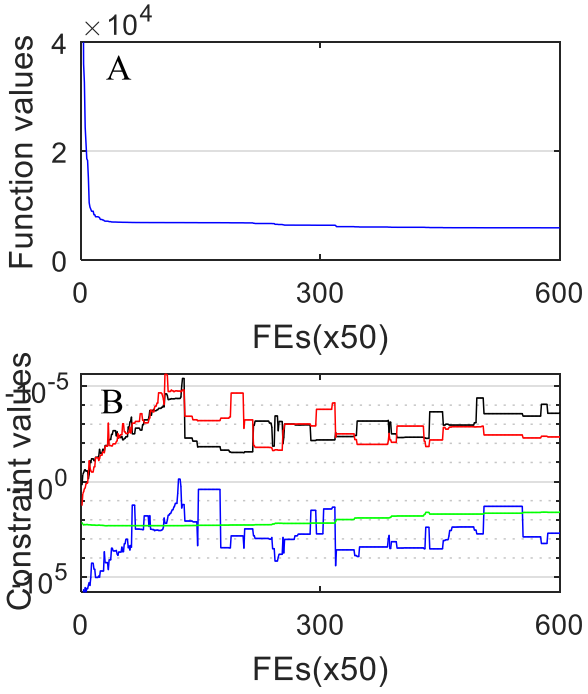

Fig. 22. (A) Function values versus FEs, and (B) constraint values versus FEs.

$(\mu + \lambda)$ES (Mezura-Montes and Coello, 2005), and CSA (Askarzadeh, 2016). Table 9 provides the two sets of results, using MRFO for 8000 FEs and 30,000 FEs, respectively. The results of MRFO are obviously better than those of PSO2 and QPSO for the same number of FEs. With the same or even less number of FEs, MRFO finds the best results in terms of the worst, mean, best and standard deviation of best-so-far solutions. The function and constraint values versus FEs are illustrated Fig. 22, respectively. Though the solutions of MRFO obtained in the early iterations satisfy all the constraints, their function values still remain very high. With a few iterations, the solutions quickly converge towards the global optimum.

4.3. Welded beam design

Tthe tension/compression spring design problem is described in Coello (2000a,b). The fabrication cost of a welded beam is to be minimized subject to seven constraints. There are four design variables to be optimized as depicted in Fig. 23.

The optimization results with 30,000 FEs in terms of decision variables, constraint values, and function values are compared using seven optimizers in Table 10. Once more, the results reveal that MRFO significantly outperforms the results of other algorithms. The results of some competitive optimizers such as GA2 (Coello, 2000a,b), GA3 (Coello and Montes, 2002), CPSO (He and Wang, 2006), HPSO (He and Wang, 2007), PSO-DE (Liu et al., 2010), WOA (Mirjalili and Lewis, 2016), EPSO (Ngo et al., 2016), ABC (Akay and Karaboga, 2012a,b), $(\mu + \lambda)$ES (Mezura-Montes and Coello, 2005), and SC (Ray and Liew, 2003) are compared with those of MRFO and the comparisons are listed in Table 11. Observing the results in Table 11, two sets of statistical

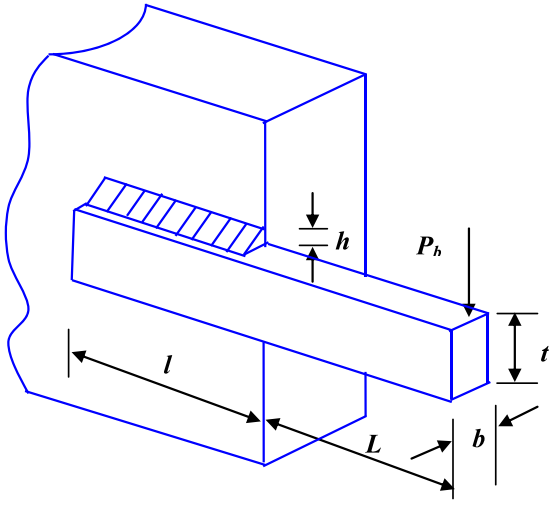

Fig. 23. Welded beam design.

Table 10
Result comparisons of seven optimizers for welded beam design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
$x_1(T_s)$	0.2057296	0.2039531	0.1818246	0.2056932	0.1991575	0.1758009	0.1649867
$x_2(T_h)$	3.4704887	3.5695905	5.7718106	3.4713620	3.7071659	5.7719073	4.9915687
$x_3(R)$	9.0366239	9.0041963	6.6114009	9.0366210	8.9907959	6.8847162	8.8119263
$x_4(L)$	0.2057296	0.2072141	0.3843461	0.2057301	0.2102505	0.3544355	0.2191593
g_1	−9.4406E−09	−144.7342173	−0.1370077	−0.2721622	−205.1531509	−0.0212231	−479.9965218
g_2	−1.2678E−08	0	−0.0098009	−0.0415918	−345.0439415	−8.5484E−06	−383.8252323
g_3	−6.2250E−13	−0.0032611	−0.2025214	−3.6890E−05	−0.0110930	−0.1786346	−0.0541726
g_4	−3.432983785	−3.4185368	−2.5794210	−3.432904573	−3.3854996	−2.6755927	−3.2326284
g_5	−0.08072964	−0.0789531	−0.0568246	−0.0806932	−0.0741575	−0.0508009	−0.0399867
g_6	−0.235540323	−0.2354882	−0.2302362	−0.2355403	−0.2356338	−0.2310208	−0.2353613
g_7	−8.0045E−09	−116.3317530	−2.5254E+04	−0.0353824	−382.8890645	−1.9282E+04	−1.1333E+03
f_3	1.7248523	1.7411391	2.6279150	1.7249164	1.7727837	2.5182363	1.9146221

Table 11
Result comparisons of optimizers in literature.

Methods	Worst	Mean	Best	Std	FEs
GA2	1.7858350	1.7719730	1.7483090	1.1200E−02	900,000
GA3	1.9934080	1.7926540	1.7282260	7.4700E−02	80,000
CPSO	1.7821430	1.7488310	1.7280240	1.2900E−02	240,000
HPSO	1.8142950	1.7490400	1.7248520	4.0100E−02	81,000
PSO-DE	1.7248520	1.7248520	1.7248520	6.7000E−16	66,600
DE	1.8241050	1.7681580	1.7334610	2.2100E−02	204,800
WOA	NA	1.7320000	NA	0.0226000	9 900
EPSO	1.7472200	1.7282190	1.7248530	5.6200E−03	50,000
ABC	NA	1.7419130	1.7248520	3.100E−02	30,000
$(\mu+\lambda)$ES	NA	1.7776920	1.7248520	8.800E−02	30,000
SC	6.3996780	3.0025883	2.3854347	9.600E−01	33,095
MRFO	1.72545088	1.7250686	1.7249121	2.2800E−01	9 900
MRFO	1.7248648	1.7248547	1.7248523	3.8320E−06	30,000

results found by MRFO for 9900 FEs and 30,000 FEs are provided. The results of MRFO are obviously better than those of WOA for the same number of FEs. It also demonstrates that the results found by MRFO for 30,000 FEs are more outstanding than those found by the others for more FEs. The function and constraint values versus FEs are illustrated in Fig. 24.

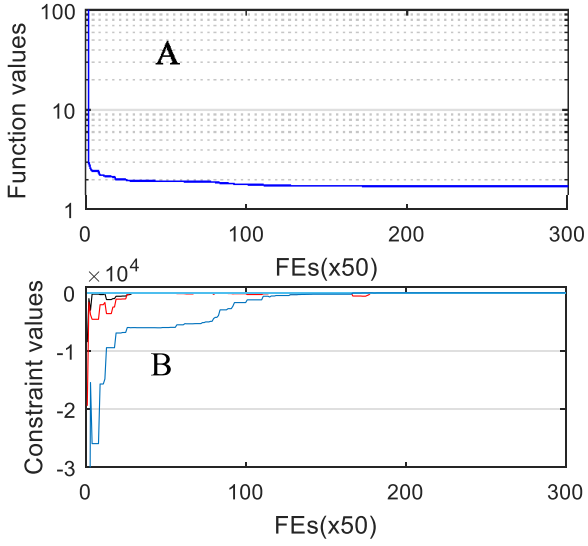

Fig. 24. (A) Function values versus FEs, and (B) constraint values versus FEs.

4.4. Speed reducer design

The speed reducer design problem was described in Mezura-Montes and Coello (2005) earlier. This case needs to achieve a minimum cost subject to eleven constraints and consists of seven design variables, as depicted in Fig. 25.

The results of seven algorithms in terms of decision variables, constraint values and function values for 25,000 FEs are presented in Table 12 for this case. According to this table, MRFO is superior to the other algorithms for the same number of FEs. In addition, Table 13 shows the results obtained from MRFO and the other reported metaheuristic optimizers, such as SC (Ray and Liew, 2003), DELC (Wang and Li, 2010), HEAA (Wang et al., 2009), PSO-DE (Liu et al., 2010), DEDS (Zhang et al., 2008), ABC (Akay and Karaboga, 2012a,b), $(\mu+\lambda)$ES (Mezura-Montes and Coello, 2005), and MDE (Montes et al., 2016). Two sets of results of MRFO algorithm for 20,000 FEs and

Table 12
Result comparisons of seven optimizers for speed reducer design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
$x_1(b)$	3.5000000	3.5658117	3.5638400	3.5000019	3.5042391	3.5342305	3.5000364
$x_2(m)$	0.7000000	0.7029491	0.7009633	0.7000000	0.7000000	0.7008723	0.7000003
$x_3(z)$	17.0000000	17.0905244	17.0972056	17.0000000	17.0000000	17.8168441	17.0000166
$x_4(l_1)$	7.3000000	7.5889560	7.5892570	7.3634690	7.3556853	7.3977776	7.3008861
$x_5(l_2)$	7.7153199	7.9429409	7.9881058	7.7296891	8.2310069	8.2454805	7.7162202
$x_6(d_1)$	3.3502147	3.4401770	3.3998036	3.3503350	3.3526213	3.4920624	3.3503003
$x_7(d_2)$	5.2866545	5.3141866	5.3871209	5.2866599	5.2883451	5.4292329	5.2866611
g_1	−0.0739153	−0.1033928	−0.0981592	−0.0739158	−0.0750356	−0.1271085	−0.0739266
g_2	−0.1979985	−0.2276392	−0.2234344	−0.1979990	−0.1989687	−0.2787217	−0.1980091
g_3	−0.4991722	−0.4986962	−0.4731107	−0.4860690	−0.4890931	−0.5792119	−0.4990417
g_4	−0.9046439	−0.8990573	−0.9025366	−0.9041105	−0.8843648	−0.9002801	−0.9046111
g_5	−1.5871E−10	−0.0760776	−0.0427391	−1.1224E−06	−0.0020587	−0.1173404	−7.5182E−05
g_6	−1.7788E−10	−0.0154332	−0.0548714	−3.1994E−07	−0.0008567	−0.0767075	−3.6049E−06
g_7	−0.7025000	−0.6996558	−0.7003872	−0.7025000	−0.7025000	−0.6878167	−0.7024996
g_8	−5.7087E−12	−0.0143211	−0.0165618	−5.3710E−07	−0.0012097	−0.0084513	−9.9622E−06
g_9	−0.7958333	−0.7911180	−0.7918233	−0.7958332	−0.7955861	−0.7935796	−0.7958311
g_{10}	−0.0513258	−0.0696658	−0.0776824	−0.0594783	−0.0580168	−0.0351030	−0.0514233
g_{11}	−9.2751E−10	−0.0248442	−0.0203143	−0.0018582	−0.0624258	−0.0452762	−0.0001157
f_4	2994.4710667	3098.7993343	3127.7366085	2995.3808600	3009.6425347	3301.5843116	2994.5431994

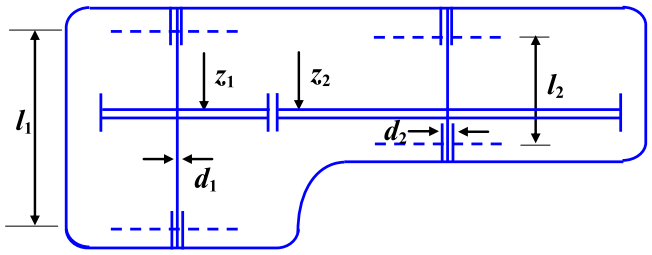

Fig. 25. Speed reducer design.

Table 13
Result comparisons of optimizers in literature.

Methods	Worst	Mean	Best	Std	FEs
SC	3009.9647360	3001.7582640	2994.7442410	4	54,456
DELC	2994.4710660	2994.4710660	2994.4710660	1.9000E−12	30,000
HEAA	2994.7523110	2994.6133680	2994.4991070	7.0000E−02	40,000
PSO-DE	2996.3482040	2996.3481740	2996.3481670	6.4000E−06	54,350
DEDS	2994.4710660	2994.4710660	2994.4710660	3.6000E−12	30,000
ABC	NA	2997.0580000	2997.0580000	0	30,000
$(\mu + \lambda)$ES	NA	2996.3480940	2996.3480940	0	30,000
MDE	NA	2996.3672200	2996.3566890	8.2000E−03	24,000
MRFO	2994.4710663	2994.4710662	2994.4710667	3.6508E−08	30,000
MRFO	2994.5247703	2994.4928459	2994.4799939	0.0146037	20,000

30,000 FEs are provided. From Table 13, the results of MRFO for 20,000 FEs is substantially better than those of MDE for 24,000 FEs. Besides, the results of MRFO for 30,000 FEs is not inferior to those of the others for more number of FEs. The function and each constraint values versus FEs for this case are depicted in Fig. 26. As shown in the figure, the obtained solutions at the beginning of iterations violate three constraints that greatly increase the function value. After 200 FEs, the obtained solutions tend to satisfy all the constraints, thus instantly decreasing the function value.

4.5. Rolling element bearing design

In this case (Rao and Tiwari, 2007; Gupta et al., 2017), we maximize the dynamic load carrying capacity of rolling element bearing. There are ten design variables and constraints, as depicted in Fig. 27.

Table 14 shows the comparisons of statistical results found by seven algorithms in terms of decision variables, constraint values and function values for 30,000 FEs. It is to be noted that MRFO provides the best solutions with considerable improvement compared to its competitors.

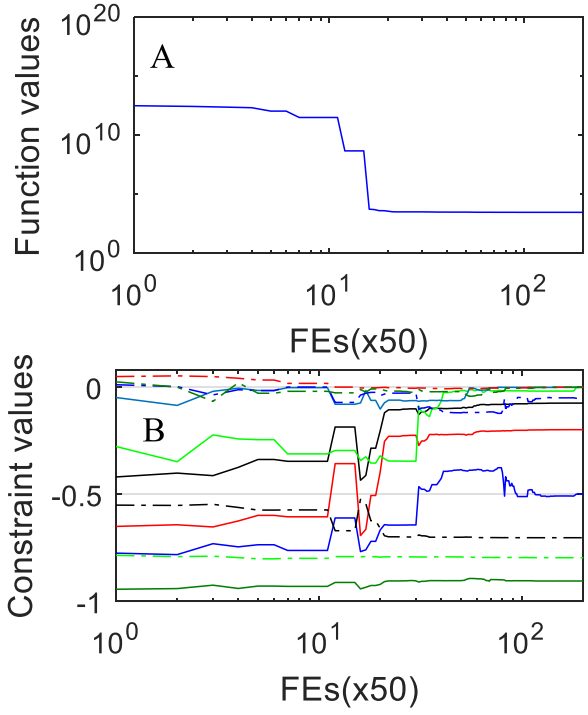

Fig. 26. (A) Function values versus FEs, and (B) constraint values versus FEs.

Table 15
Result comparisons of optimizers in literature.

Methods	Worst	Mean	Best	Std	FEs
GA4	NA	NA	81 843.3000	NA	225,000
ABC	78 897.8100	81 496.0000	81 859.7416	0.6900	10,000
TLBO	80 807.8551	81 438.9870	81 859.7400	0.6600	10,000
MBA	84 440.1948	85 321.4030	85 535.9611	211.5200	15,100
MRFO	77 452.7627	83 927.5741	85 537.2111	2799.3392	10,000
MRFO	77 455.2949616	83 930.3612323	85 549.239	2412.6229000	30,000

This case was tackled using some metaheuristic optimizers such as GA4 (Rao and Tiwari, 2007), TLBO (Rao et al., 2011), ABC (Akay

Table 14
Result comparisons of seven optimizers for rolling element bearing design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
D_m	125.7190556	125.4108239	128.4682013	125.7190604	125.4427870	126.0663153	125.7165666
D_b	21.4255902	20.5762896	20.6129842	21.4255668	21.2051590	21.0307643	21.4243043
Z	11	11	11	11	11	10	11
f_i	0.5150000	0.5159001	0.5182942	0.5150000	0.5150000	0.5150000	0.5150003
f_o	0.5150000	0.5245671	22.3060227	0.5504475	0.5416852	0.5461764	0.5231184
K_{Dmin}	0.4050856	0.4547713	0.3773456	0.4628408	0.5000000	0.4679413	0.4639528
K_{Dmax}	0.6905778	0.6080293	10.3683602	0.6132153	0.7000000	0.6462514	0.6642760
e	0.3000000	0.3260293	0.2497668	0.3000000	0.3000000	0.3035991	0.3000578
ζ	0.0536602	0.0236063	13.6845643	0.0850911	0.0975781	0.0716772	0.0477450
ϵ	0.6925802	0.6644673	0.5203179	0.6307493	0.6015492	0.6682039	0.6963909
g_1	2.8493E−11	0.3092442	0.5474579	9.2333E−06	0.0618292	1.1804414	0.0002860
g_2	14.4951855	9.3185891	14.8117746	10.4522771	7.4103180	9.3056386	10.3719097
g_3	5.4892668	1.4094689	684.5592483	0.0739387	6.5896820	3.1760721	3.6507118
g_4	14.1341179	6.3124058	3424.6092790	21.9918280	24.8373069	18.9856122	12.6528103
g_5	12.6960067	5.4907581	3417.6728765	20.5537072	23.9517329	16.8529815	11.2196771
g_6	0.7190556	0.4108239	3.4682013	0.7190604	0.4427870	1.0663153	0.7165666
g_7	6.9188E−09	0.2979702	0.3109674	1.6338E−05	0.3144793	0.0665390	0.0010346
g_8	−0.6481831	−0.6422693	−5.0034485	−2.5030864	−3.1586822	−0.9846479	−0.5325768
g_9	3.2131E−12	0.0009001	0.0032942	0	0	3.9989E−08	3.3705E−07
g_{10}	1.3308E−08	0.0095671	21.7910227	0.0354475	0.0266852	0.0311764	0.0081184
f_5	85 549.239	77 797.351	73 577.819	85 541.451	83 988.259	77 682.899	85 537.805

Table 16
Result comparisons of seven optimizers for multiple disc clutch brake design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
r_i	70	77	72	71	79	72	71
r_o	90	98	92	92	99	92	91
t	1	1	1	1	1	2	1
F	835	736	918	835	729	815	849
Z	3	3	3	3	3	3	3
g_1	0	1	0	1	0	0	0
g_2	24	24	24	24	24	22	24
g_3	0.9164436	0.9359049	0.9107180	0.9228171	0.9347286	0.9213930	0.9164928
g_4	9.8240885	9.8524698	9.8073832	9.8344063	9.8472767	9.8304134	9.8220170
g_5	7.8946965	7.6982598	7.8426028	7.8545299	7.6601802	7.8426028	7.8686513
g_6	1.19797141	0.6856600	2.6637714	1.2904716	0.7245325	1.0372174	1.5177828
g_7	41.3250000	37.5912000	53.7209756	42.0288957	37.8651124	40.1239024	43.7996914
g_8	13.8028592	14.3143400	12.3362286	13.7095284	14.2754675	13.9627826	13.4822172
f_6	0.3136566	0.3602150	0.3214980	0.3355146	0.3489430	0.4822470	0.3175773

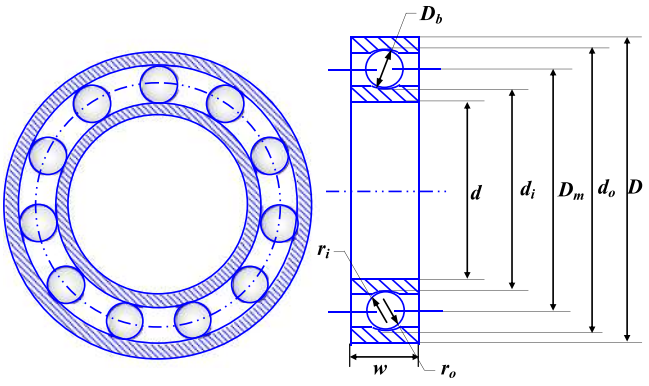

Fig. 27. Rolling element bearing design.

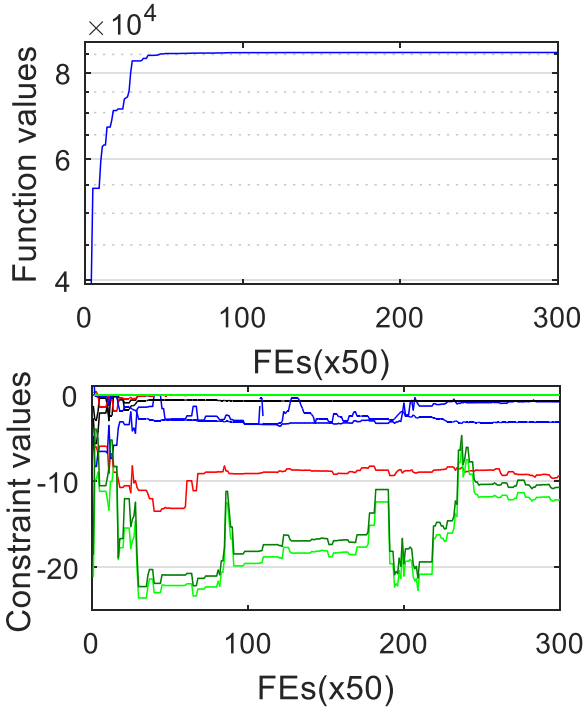

Fig. 28. (A) Function values versus FEs, and (B) constraint values versus FEs.

and Karaboga, 2012a,b), and MBA (Sadollah et al., 2013). The results of these approaches and MRFO for two different number of FEs are compared in Table 15. With the same number of FEs, MRFO

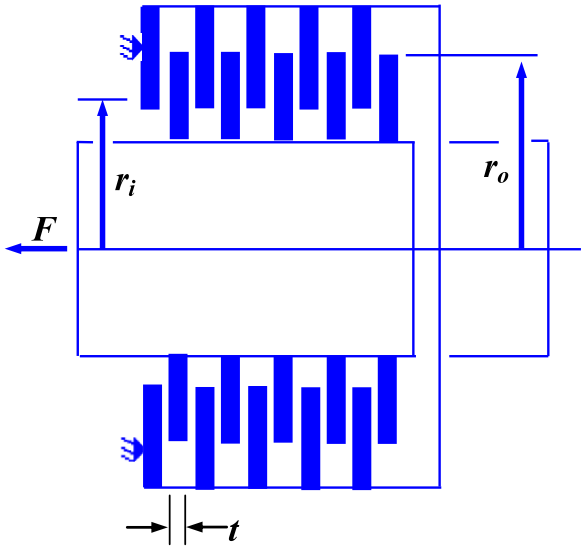

Fig. 29. Multiple disc clutch brake design.

outperforms ABC and TLBO in terms of the mean and best of best-so-far solutions. Moreover, it clearly offers a better best-so-far solution using less number of FEs than MBA and GA4. The function and each constraint values versus FEs for this case are provided in Fig. 28.

4.6. Multiple disc clutch brake design

The intent of this problem (Osyczka, 2002) is to minimize the mass of the multiple disc clutch brake. There are five discrete decision variables which need to be optimized as depicted in Fig. 29, including inner radius (r_i = 60, 61, 62, ...,80), outer radius (r_o = 90, 91, 92, ..., 110), thickness of the disc (t = 1, 1.5, 2, 2.5, 3), actuating force (F = 600, 610, 620, ..., 1000), and number of friction surfaces (Z = 2, 3, 4, 5, 6, 7, 8, 9).

For this problem, Table 16 shows the comparisons of seven algorithms in terms of decision variables, constraint values, and function values for 1000 FEs. Though both MRFO and ABC provide different optimal solutions, they have the same function value which is significantly superior to those of the others. The optimum solution obtained by MRFO is r_i = 70 mm, r_o = 90 mm, t = 1 mm, F = 933 N, and Z = 3.

The comparisons of the reported optimizers including TLBO and ABC (Rao et al., 2011) are illustrated in Table 17. As the table shows, with less computational efforts, MRFO offers its best results in terms of the mean, best, and standard deviation of best-so-far solutions. Fig. 30 shows the function values of the two optimizers and MRFO versus FEs. According to Fig. 30(A), the convergence rate of TLBO is faster

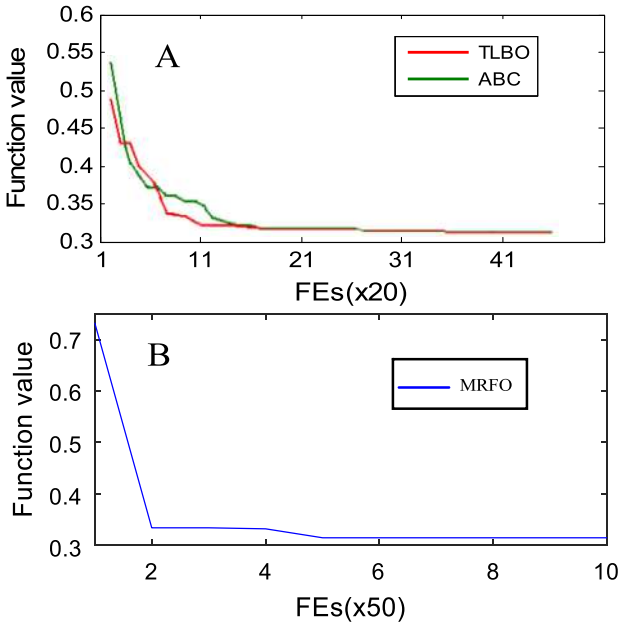

Fig. 30. (A) Convergence curves of TLBO and ABC, and (B) convergence curve of MRFO.

Table 17
Result comparisons of optimizers in literature.

Methods	Worst	Mean	Best	Std	FEs
ABC	0.352864	0.324751	0.313657	0.54	600
TLBO	0.392071	0.3271662	0.313657	0.67	600
MRFO	0.3332601	0.3243226	0.3136566	0.0054725	500

than ABC in earlier iterations, but with the increase of iterations, the convergence of both algorithms becomes nearly the same (Rao et al., 2011). From Fig. 30(B), MRFO obtains faster convergence rate than the two optimizers that provide the optimal solution after about 600 FEs. However, MRFO can offer the same optimal solution only using 500 FEs.

4.7. Belleville spring design

This case (Coello, 2000a,b) is minimization of weight while satisfying a number of complex constraints. There are four decision variables and seven constraints, as depicted in Fig. 31.

This problem is solved by seven optimizers for 50,000 FEs, the comparisons in terms of decision variables, constraint values and function values are shown in Table 18. As shown in the table, MRFO is again highly competitive with the other heuristic approaches.

This problem was also studied by GA5 (Coello, 2000a,b), ABC (Rao et al., 2011), and TLBO (Rao et al., 2011). Table 19 represents the

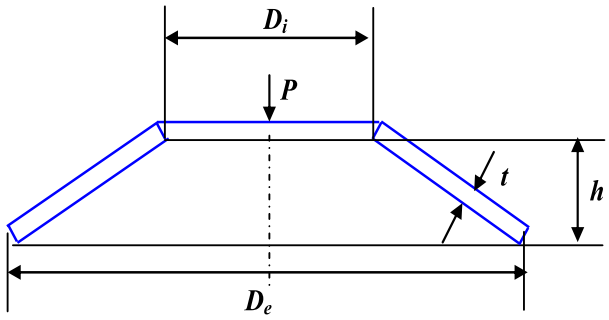

Fig. 31. Belleville spring design.

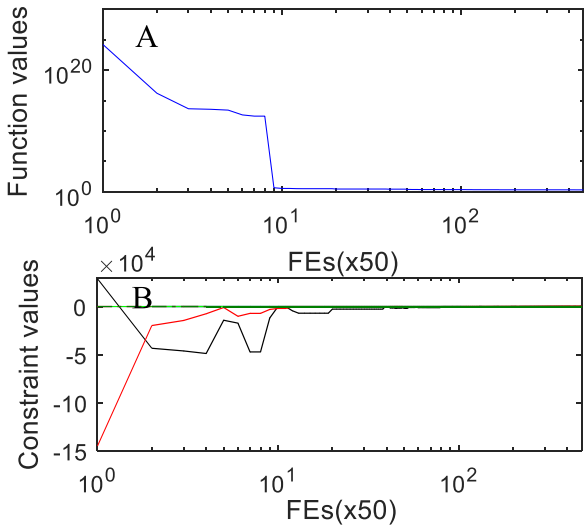

Fig. 32. (A) Function values versus FEs, and (B) constraint values versus FEs.

Table 19
Result comparisons of optimizers in literature.

Methods	Worst	Mean	Best	Std	FEs
ABC	2.1042970	1.9954750	1.9796750	0.07	150,000
TLBO	1.9797570	1.9796875	1.9796750	0.45	150,000
GA5	NA	NA	2.1219640	NA	24,000
MRFO	1.9796885	1.9796782	1.9796747	5.7748E−06	130,000
MRFO	2.0911527	2.0257815	1.9815209	0.0033092	24,000

comparisons of different optimizers for the best obtained solution. From Table 19, compared with GA5, with the same number of FEs, the better best-so-far solution is obtained using MRFO with $t = 0.2042251$, $h = 0.2000150$, $D_i = 10.0292262$, and $D_o = 12.0099244$. Additionally, compared with TLBO and ABC, MRFO evidently offers its best results in

Table 18
Result comparisons of seven optimizers for Belleville spring design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
t	0.2042234	0.2054955	0.2064078	0.2059945	0.2387035	0.1589206	0.2062420
h	0.2000546	0.2072268	0.2016670	0.2000000	0.2243158	0.4423810	0.2020973
D_i	10.0144429	9.5095439	8.7777665	8.8765657	6.7126918	6.9817740	9.6994069
D_e	11.9983046	11.6526267	11.0630741	11.1215669	10.2131069	11.7779728	11.7747325
g_1	36.2169053	0.0040697	55.3976306	2.9020E−06	1.1364E+03	1.4285E+04	166.5444890
g_2	2.1460124	0.0099363	1.6710528	1.7073E−05	1.3743E+03	8.4389E−05	81.0822224
g_3	5.4557E−05	0.0072268	0.0016670	6.1801E−11	0.0243158	0.0211905	0.0020973
g_4	1.5957220	1.5872777	1.5919252	1.5940055	1.5369807	1.3986984	1.5916607
g_5	0.0116954	0.3573733	0.9469259	0.8884331	1.7968931	0.2320272	0.2352675
g_6	1.9838618	2.1430828	2.2853076	2.2450012	3.5004151	4.7961988	2.0753256
g_7	0.1991590	0.2033043	0.2117550	0.2109132	0.2359174	0.2077643	0.2026190
f_7	1.9822916	2.0714656	2.0802102	2.0555969	3.1434302	3.1781964	2.0429395

Table 20
Result comparisons of seven optimizers for hydrostatic thrust bearing design.

	MRFO	PSO	GA	DE	CS	GSA	ABC
R	5.9647493	6.1896857	10.4790564	6.8408768	6.1084093	7.6938952	6.5305485
R_o	5.3989193	5.6453752	10.0361401	6.3535608	5.4921343	7.2072523	5.8080458
μ	5.3609E−06	5.9915E−06	5.8974E−06	9.0310E−06	6.0291E−06	8.5104E−06	5.8511E−06
Q	2.2769653	2.989664829	7.840476052	13.58961281	3.124919379	12.6978065	3.6094940
g_1	0.0005970	31.79166896	1.4552E−11	1.46E−11	3.7195E+03	9.3178E+03	4.0368E+03
g_2	3.3247221	80.9615770	694.4040306	260.9951073	8.2773890	367.1936101	120.5370892
g_3	0.0350159	9.3640171	8.0558642	41.2005477	9.8792476	36.8267640	7.4016918
g_4	0.0003260	0.0005076	0.0013195	0.0018619	0.0005685	0.0017724	0.0007520
g_5	0.5658300	0.5443105	0.4429163	0.4873159	0.6162750	0.4866429	0.7225028
g_6	0.0009963	0.0009956	0.0009867	0.0009881	0.0009958	0.0009881	0.000995464
g_7	0.0349563	7.8111418	1.4619E+03	1.9336E−09	337.4342952	157.5439944	1.2495E+03
f_8	1628.8240216	1777.943014	4212.608791	2623.644245	1866.190218	2954.18062	2214.134876

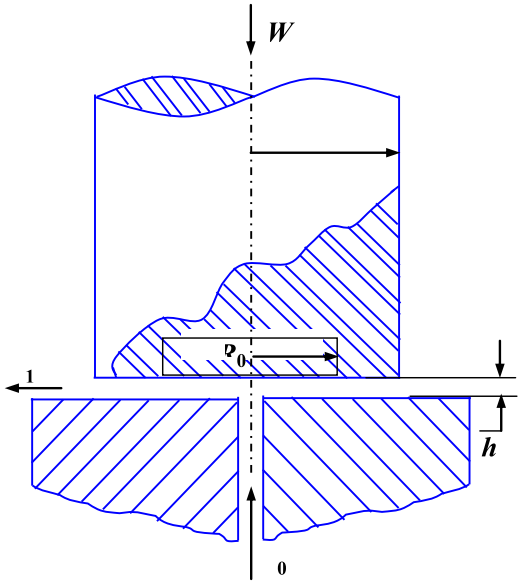

Fig. 33. Hydrostatic thrust bearing design.

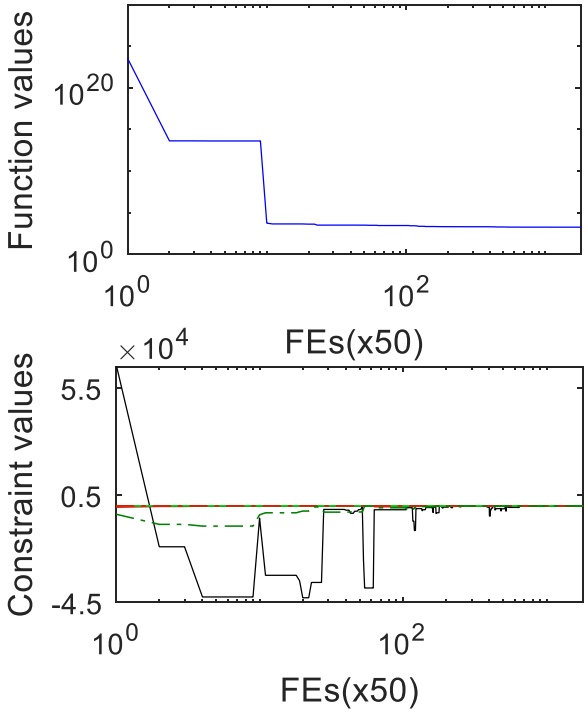

Fig. 34. (A) Function values versus FEs, and (B) constraint values versus FEs.

Table 21
Result comparisons of optimizers in literature.

Methods	Worst	Mean	Best	Std	FEs
IPSO	NA	1757.3768400	1632.2149	16.851024	90,000
GASO	NA	NA	1950.2860	NA	16,000
GeneAS	NA	NA	2161.6	NA	NA
BGA	NA	NA	2295.1	NA	NA
MRFO	2277.5687497	2031.5436592	1792.8179135	284.1747456	16,000
MRFO	2101.3816223	1749.2715233	1626.4216806	28.8128424	90,000

terms of the worst, best and standard deviation of best-so-far solutions with less number of FEs, and the best obtained function value is 1.9796747 with t = 0.2041434, h = 0.2, D_i = 10.0304732, and D_o = 12.0099999. The function and constraint values versus FEs for this case are demonstrated in Fig. 32, obviously, MRFO tends to approximate the optimal solution in a short period of time.

4.8. Hydrostatic thrust bearing design

The last utilized engineering case, proposed by Siddall (1982), is the hydrostatic thrust bearing design problem. This problem, depicted in Fig. 33, is the minimization of power loss of the bearing while satisfying seven constraints. This case has four continuous decision variables to be optimized.

The statistical results, given by seven algorithms in terms of decision variables, constraint values and function values for 50,000 FEs, are presented in Table 20 for this case. As observed in the table, MRFO is highly competitive with other heuristic approaches. This problem was previously handled by IPSO (He et al., 2004), GASO (Coello, 2000a,b), GeneAS (Deb and Goyal, 1997), and BGA (Deb and Goyal, 1997) with varying degrees of success, and the results offered by them are listed in Table 21. For comparison, this problem is handled using MRFO with 90,000 FEs and 16,000 FEs, respectively. From Table 21, compared with GASO, MRFO produces better results with the same number of FEs for the best function value f_8 = 1626.4216806 with R = 6.0278512, R_o = 5.4681520, μ = 5.3662E−06, and Q = 2.2755966. Compared with IPSO, MRFO produces better results with the same number of FEs for the best function value f_8 = 1792.8179135 with R = 5.9562696, R_o = 5.38955360, μ = 6.6246E−06, and Q = 3.5584835. Obviously, with the same computational efforts, MRFO provides more satisfactory results than its counterparts. Fig. 34 shows the function and constraint values versus FEs for the hydrostatic thrust bearing design problem.

The convergence curves of seven algorithms, based on the average best-so-far of the 20 runs, are presented in Fig. 35 to evaluate the convergence rate of MRFO for constrained engineering problems. As illustrated in this figure, in earlier iterations, the individuals in MRFO are inclined to extensively explore the entire search space to position promising areas. With the increase of iterations, the region with the optimum is intensively exploited. The most evident difference between

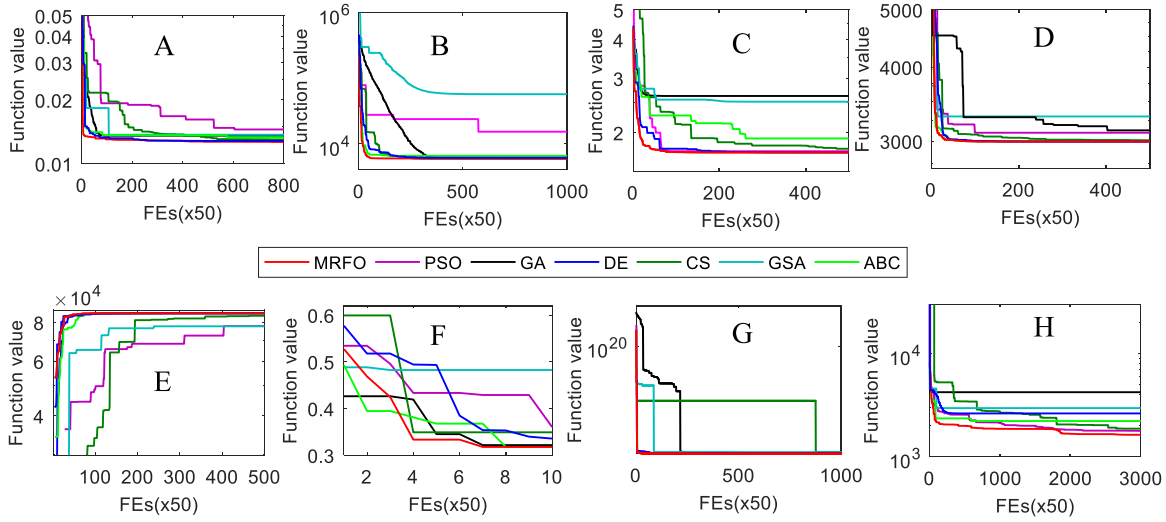

Fig. 35. Comparisons of convergence curves of seven algorithms for eight engineering problems, (A) Tension/compression spring design, (B) pressure vessel design, (C) welded beam design, (D) speed reducer design, (E) rolling element bearing design, (F) multiple disc clutch brake design, (G) Belleville spring design, (H) hydrostatic thrust bearing design.

MRFO and other optimizers is that MRFO offers a faster convergence rate in the entire optimization process and obtains competitive solutions with less computational efforts for the majority of engineering problems.

5. Conclusions

In terms of the intelligent foraging behaviors of manta rays, a new optimization approach, named Manta Rays Foraging Optimization (MRFO), is proposed in this study. This algorithm has three foraging operators to mimic manta rays' hunt for food, including chain foraging, cyclone foraging, and somersault foraging. This approach, with few adjustable parameters, is easy to implement, which in turn makes it very potential for applications in many engineering fields. A diverse set of benchmark functions including unimodal, multimodal, low-dimensional and composition functions are used to confirm the performance of MRFO from different aspects. The comparisons show MRFO is often superior to other well-known competitors.

In order to verify its ability to solve real-world problems, an extensive study is conducted on eight real-world engineering problems such as tension/compression spring design, pressure vessel design, etc. The comparisons suggest that MRFO is more effective than other well-known optimizers. The results of benchmark functions and engineering problems also reveal MRFO has powerful global optimization ability not only on unconstrained problems but also on constrained problems. It is very suitable for handling real-world problems requiring less computational expense with a specified precision for final solutions.

The novelty and contribution of this work are highlighted as follow.

(1) A novel nature-inspired manta ray foraging optimization algorithm is developed in this work. The foraging behaviors of manta rays are investigated thoroughly and formulated mathematically including every characteristic of their foraging behaviors.

(2) There are three special search strategies which play three different roles in the proposed algorithm. The chain foraging behavior significantly contributed to the local search ability of the algorithm; the cyclone foraging behavior of manta rays is greatly dedicated to the global search ability of the algorithm; the somersault foraging behavior enhances the local search ability and raises the convergence rate.

(3) The experimental results on complex benchmark problems indicate that MRFO is very effective and reliable. The optimization results of engineering designs demonstrate that the MRFO optimizer can lead to promising improvement on solution precision with less computation cost compared with other well-established optimizers.

For future work, the binary MRFO may be presented to deal with complex discrete problems. The extended MRFO could also be tailored to address multi-objective optimization.

Table A.1
Unimodal test functions.

Name	Function	D	Range	f_{opt}
Sphere	$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	$[-100,100]^n$	0
Schwefel 2.22	$f_2(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	30	$[-10,10]^n$	0
Schwefel 1.2	$f_3(x)=\sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	$[-100,100]^n$	0
Schwefel 2.21	$f_4(x)=\max_i \{\lvert x_i\rvert, 1\le i\le n\}$	30	$[-100,100]^n$	0
Rosenbrock	$f_5(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i)^2)+(x_i-1)^2$	30	$[-30,30]^n$	0
Step	$f_6(x)=\sum_{i=1}^{n}(x_i+0.5)^2$	30	$[-100,100]^n$	0
Quartic	$f_7(x)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	30	$[-1.28,1.28]^n$	0

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.103300.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (11972144), Natural Science Foundation of Hebei Province of China (E2018402092, F2017402142), and Scientific Research Key Project of University of Hebei Province of China (ZD2017017).

Appendix A

See Tables A.1–A.4.

Appendix B

B.1. Tension/compression spring design

Consider variable $\vec{x}=[x_1,x_2,x_3]=[d,D,N]$.

Minimize $f_1(\vec{x})=(x_3+2)x_2x_1^2$.

Subject to $g_1(\vec{x})=1-\frac{x_3x_2^3}{71785x_1^4}\le 0$,

Table A.2
Multimodal test functions.

Name	Function	D	Range	f_{opt}
Schwefel	$f_8(x) = -\sum_{i=1}^{n}(x_i \sin(\sqrt{\lvert x_i \rvert}))$	30	$[-500,500]^n$	−12 569.5
Rastrigin	$f_9(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)^2$	30	$[-5.12,5.12]^n$	0
Ackley	$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$	30	$[-32,32]^n$	0
Griewank	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos(\frac{x_i-100}{\sqrt{i}}) + 1$	30	$[-600,600]^n$	0
Penalized	$f_{12}(x) = \frac{\pi}{n}10\sin 2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin 2(\pi y_i + 1)]$ $+(y_n - 1)^2 + \sum_{i=1}^{30} u(x_i, 10, 100, 4)$	30	$[-50,50]^n$	0
Penalized2	$f_{13}(x) = 0.1\sin 2(3\pi x_1) + \sum_{i=1}^{29}(x_i - 1)^2 p[1 + \sin 2(3\pi x_{i+1})]$ $+(x_n - 1)^2[1 + \sin 2(2\pi x_{30})] + \sum_{i=1}^{30} u(x_i, 5, 10, 4)$	30	$[-50,50]^n$	0

Table A.3
Low-dimensional multimodal test functions.

Name	Function	D	Range	f_{opt}
Foxholes	$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	$[-65.536,\ 65.536]^n$	0.998
Kowalik	$f_{15}(x) = \sum_{i=1}^{11}\left\lvert a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right\rvert^2$	4	$[-5,\ 5]^n$	3.075×10^{-4}
Six Hump Camel	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,\ 5]^n$	−1.0316
Branin	$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	$[-5,\ 10] \times [0,\ 15]$	0.398
GoldStein–Price	$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$ $\times\left[30 + (2x_1 + 1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	$[-2,\ 2]^n$	3
Hartman 3	$f_{19}(x) = -\sum_{i=1}^{4}\exp\left[-\sum_{j=1}^{3}a_{ij}(x_j - p_{ij})^2\right]$	3	$[0,\ 1]^n$	−3.86
Hartman 6	$f_{20}(x) = -\sum_{i=1}^{4}\exp\left[-\sum_{j=1}^{6}a_{ij}(x_j - p_{ij})^2\right]$	6	$[0,\ 1]^n$	−3.322
Shekel 5	$f_{21}(x) = -\sum_{i=1}^{5}\left\lvert(x_i - a_i)(x_i - a_i)^T + c_i\right\rvert^{-1}$	4	$[0,\ 10]^n$	−10.1532
Shekel 7	$f_{22}(x) = -\sum_{i=1}^{7}\left\lvert(x_i - a_i)(x_i - a_i)^T + c_i\right\rvert^{-1}$	4	$[0,\ 10]^n$	−10.4028
Shekel 10	$f_{23}(x) = -\sum_{i=1}^{10}\left\lvert(x_i - a_i)(x_i - a_i)^T + c_i\right\rvert^{-1}$	4	$[0,\ 10]^n$	−10.5363

Table A.4
Composition test functions.

Function	Name	D	Range	f_{opt}
$f_{24}(x)$	Composition function 1 ($N = 5$)	30	$[-100,100]^n$	2300
$f_{25}(x)$	Composition function 2 ($N = 3$)	30	$[-100,100]^n$	2400
$f_{26}(x)$	Composition function 3 ($N = 3$)	30	$[-100,100]^n$	2500
$f_{27}(x)$	Composition function 4 ($N = 5$)	30	$[-100,100]^n$	2600
$f_{28}(x)$	Composition function 5 ($N = 5$)	30	$[-100,100]^n$	2700
$f_{29}(x)$	Composition function 6 ($N = 5$)	30	$[-100,100]^n$	2800
$f_{30}(x)$	Composition function 7 ($N = 3$)	30	$[-100,100]^n$	2900
$f_{31}(x)$	Composition function 8 ($N = 3$)	30	$[-100,100]^n$	3000

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0,$$
$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0,\ g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0.$$

Variable range $0.05 \le x_1 \le 2, 0.25 \le x_2 \le 1.3, 2 \le x_3 \le 15$.

B.2. Pressure vessel design

Consider variable $\vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L]$.

Minimize $f_2(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$.

Subject to $g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0,\ g_2(\vec{x}) = -x_2 + 0.00954x_3 \le 0,$
$g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0,\ g_4(\vec{x}) = x_4 - 240 \le 0.$

Variable range $0 \le x_1 \le 99, 0 \le x_2 \le 99, 10 \le x_3 \le 200, 10 \le x_4 \le 200$.

B.3. Welded beam design

Consider variable $\vec{x} = [x_1, x_2, x_3, x_4] = [h, l, t, b]$.

Minimize $f_3(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$.

Subject to

$g_1(\vec{x}) = \tau(\vec{x}) + \tau_{\max} \le 0,\ g_2(\vec{x}) = \sigma(\vec{x}) + \sigma_{\max} \le 0,\ g_3(\vec{x}) = \delta(\vec{x}) + \delta_{\max} \le 0,$
$g_4(\vec{x}) = x_1 - x_4 \le 0,\ g_5(\vec{x}) = P - P_c(\vec{x}) \le 0, g_6(\vec{x}) = 0.125 - x_1 \le 0,$
$g_7(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0.$

where

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\ \tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},$$
$$M = P(L + \frac{x_2}{2}),$$
$$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2},\ J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^3}{4} + (\frac{x_1 + x_3}{2})^2\right]\right\},$$
$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2},\ \delta(\vec{x}) = \frac{4PL^3}{Ex_4x_3^3},$$
$$P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right).$$

where $P = 6000\,\text{lb}$, $L = 14\,\text{in}$, $E = 30 \times 10^6\,\text{psi}$, $G = 12 \times 10^6\,\text{psi}$, $\tau_{\max} = 13\,600\,\text{psi}$, $\sigma_{\max} = 30\,000\,\text{psi}$, $\delta_{\max} = 0.25\,\text{in}$.

Variable range $0.1 \le x_1 \le 2,\ 0.1 \le x_2 \le 10,\ 0.1 \le x_3 \le 10, 0.1 \le x_4 \le 2$.

B.4. Speed reducer design

Consider variable $\vec{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7] = [b, m, z, l_1, l_2, d_1, d_2]$.

Minimize

$$\text{Minimize } f_4(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 0.7854x_1(x_4x_6^2 - x_5x_7^2).$$

$$\text{Subject to } g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0, g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0,$$

$$g_3(\vec{x}) = \frac{1.93x_4^2}{x_2x_6^4x_3} - 1 \le 0, g_4(\vec{x}) = \frac{1.93x_5^2}{x_2x_7^4x_3} - 1 \le 0,$$

$$g_5(\vec{x}) = \frac{((\frac{745x_4}{x_2x_3})^2 + 16.9\times10^6)^{0.5}}{110x_6^3} - 1 \le 0,$$

$$g_6(\vec{x}) = \frac{((\frac{745x_5}{x_2x_3})^2 + 157.5\times10^6)^{0.5}}{85x_7^3} - 1 \le 0,$$

$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \le 0, g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \le 0,$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \le 0, g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0,$$

$$g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0.$$

Variable range $2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,\ 17 \le x_3 \le 28,$
$7.3 \le x_4 \le 8.3,$
$7.3 \le x_5 \le 8.3,\ 2.9 \le x_6 \le 3.9,\ 5.0 \le x_7 \le 5.5.$

B.5. Rolling element bearing design

Consider variable $\vec{x} = [D_m, D_b, Z, f_i, f_o, K_{D\min}, K_{D\max}, \varepsilon, e, \zeta]$.

$$\text{Maximize } \begin{cases} f_5(\vec{x}) = f_c Z^{2/3} D_b^{1.8} & if\ D_b \le 25.4\ \text{mm} \\ f_5(\vec{x}) = 3.647 f_c Z^{2/3} D_b^{1.4} & if\ D_b > 25.4\ \text{mm}. \end{cases}$$

$$\text{Subject to } g_1(\vec{x}) = \frac{\phi_o}{2\sin^{-1}(D_b/D_m)} - Z + 1 \ge 0,$$

$$g_2(\vec{x}) = 2D_b - K_{D\min}(D - d) \ge 0,$$

$$g_3(\vec{x}) = K_{D\max}(D - d) - 2D_b \ge 0,\ g_4(\vec{x}) = D_m - (0.5 - e)(D + d) \ge 0,$$

$$g_5(\vec{x}) = (0.5 + e)(D + d) - D_m \ge 0,\ g_6(\vec{x}) = D_m - 0.5(D + d) \ge 0,$$

$$g_7(\vec{x}) = 0.5(D - D_m - D_b) - \varepsilon D_b \ge 0,\ g_8(\vec{x}) = \zeta B_w - D_b \le 0,$$

$$g_9(\vec{x}) = f_i \ge 0.515, g_{10}(\vec{x}) = f_o \ge 0.515.$$

where

$$f_c = 37.91\left[1 + \left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o - 1)}{f_o(2f_i - 1)}\right)^{0.4}\right\}^{10/3}\right]^{-0.3} \times \left(\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{f_o(1+\gamma)^{\frac{1}{3}}}\right)\left(\frac{2f_i}{2f_i - 1}\right)^{0.41},$$

$$\gamma = \frac{D_b\cos\alpha}{D_m},\ f_i = \frac{r_i}{D_b},\ f_o = \frac{r_o}{D_b},$$

$$\phi_o = 2\pi - 2 \times \cos^{-1}\frac{\{(D-d)/2 - 3(T/4)\}^2 + \{D/2 - (T/4) - D_b\}^2 - \{d/2 + (T/4)\}^2}{2\{(D-d)/2 - 3(T/4)\}\{D/2 - (T/4) - D_b\}},$$

,

$$T = D - d - 2D_b,\ D = 160,\ d = 90,\ B_w = 30,\ r_i = r_o = 11.033.$$

Variable range

$0.5(D + d) \le D_m \le 0.6(D + d),\ 0.15(D - d) \le D_b \le 0.45(D - d),$
$4 \le Z \le 50,$
$0.515 \le f_i \le 0.6,\ 0.515 \le f_o \le 0.6,\ 0.4 \le K_{D\min} \le 0.5,$
$0.6 \le K_{D\max} \le 0.7,$
$0.3 \le \varepsilon \le 0.4,\ 0.02 \le e \le 0.1,\ 0.6 \le \zeta \le 0.85.$

B.6. Multiple disc clutch brake design

Consider variable $\vec{x} = [r_i, r_o, t, F, Z]$.

$$\text{Minimize } f(\vec{x}) = \pi(r_0^2 - r_i^2)t(Z + 1)\rho.$$

$$\text{Subject to } g_1(\vec{x}) = r_o - r_i - \Delta r \ge 0, g_2(\vec{x}) = l_{\max} - (Z + 1)(t + \delta) \ge 0,$$

$$g_3(\vec{x}) = p_{\max} - p_{rz} \ge 0,\ g_4(\vec{x}) = p_{\max}v_{sr\max} - p_{rz}v_{sr} \ge 0,$$

$$g_5(\vec{x}) = v_{sr\max} - v_{sr} \ge 0,\ g_6(\vec{x}) = T_{\max} - T \ge 0,\ g_7(\vec{x}) = M_h - sM_s \ge 0,$$

$$g_8(\vec{x}) = T \ge 0.$$

where

$$M_h = \frac{2}{3}\mu F Z\frac{r_o^3 - r_i^3}{r_o^2 - r_i^2},\ p_{rz} = \frac{F}{\pi(r_o^2 - r_i^2)},\ v_{sr} = \frac{2\pi n(r_o^3 - r_i^3)}{90(r_o^2 - r_i^2)},$$

$$T = \frac{I_z\pi n}{30(M_h + M_f)},$$

$\Delta r = 20\,\text{mm}, l_{\max} = 30\,\text{mm}, v_{sr\max} = 10\,\text{m/s}, \delta = 0.5, s = 1.5, M_s = 40\,\text{N m},$
$M_f = 3\,\text{N m},$
$n = 250\,\text{rpm}, p_{\max} = 1\,\text{MPa}, I_z = 55\,\text{kg mm}^2, T_{\max} = 15\,\text{s}, F_{\max} = 1000\,\text{N},$
$r_{i\min} = 60\,\text{mm},$
$r_{i\max} = 80\,\text{mm}, r_{o\min} = 90\,\text{mm}, r_{o\max} = 110\,\text{mm}, t_{\min} = 1\,\text{mm}, t_{\max} = 3\,\text{mm},$
$F_{\min} = 600,$
$F_{\max} = 1100, Z_{\min} = 2, Z_{\max} = 9.$

B.7. Belleville spring design

Consider variable $\vec{x} = [t, h, D_i, D_e]$.

$$\text{Minimize: } f_7(\vec{x}) = 0.07075\pi\left(D_e^2 - D_i^2\right)t.$$

Subject to

$$g_1(\vec{x}) = S - \frac{4E\delta_{\max}}{(1-\mu^2)\alpha D_e^2}\left[\beta(h - \frac{\delta_{\max}}{2}) + \gamma t\right] \ge 0,$$

$$g_2(\vec{x}) = \left(\frac{4E\delta_{\max}}{(1-\mu^2)\alpha D_e^2}\left[(h - \frac{\delta}{2})((h-\delta)t + t^3)\right]\right)_{\delta=\delta_{\max}} - P_{\max} \ge 0,$$

$$g_3(\vec{x}) = \delta_1 - \delta_{\max} \ge 0, g_4(\vec{x}) = H - h - \text{t} \ge 0,$$

$$g_5(\vec{x}) = D_{\max} - D_e \ge 0, g_6(\vec{x}) = D_e - D_i \ge 0, g_7(\vec{x}) = 0.3 - \frac{h}{D_e - D_i} \ge 0.$$

where $\alpha = \frac{6}{\pi\ln K}\left(\frac{K-1}{K}\right)^2$, $\beta = \frac{6}{\pi\ln K}\left(\frac{K-1}{\ln K} - 1\right)$, $\gamma = \frac{6}{\pi\ln K}\left(\frac{K-1}{2}\right)$,
$P_{\max} = 5400\,\text{lb}$, $\delta_{\max} = 0.2\,\text{in}$, $S = 200\,000\,\text{psi}$, $E = 30\times10^6\,\text{psi}$, $\mu = 0.3$,
$H = 2\,\text{in}$, $D_{\max} = 12.01\,\text{in}$, $K = D_e/D_i$, $\delta_1 = f(a)h$, $a = h/t$.
Values of $f(a)$ vary as shown in Table B.1.

Variable range $0.01 \le t \le 6,\ 0.05 \le h \le 0.5,\ 5 \le D_i \le 15,\ 5 \le D_o \le 15.$

B.8. Hydrostatic thrust bearing design

Consider variable $\vec{x} = [R, R_o, \mu, Q]$.

$$\text{Minimize } f_8(\vec{x}) = \frac{QP_o}{0.7} + E_f.$$

Table B.1
Variation of $f(a)$ with a.

a	≤1.4	1.5	1.6	1.7	1.8	1.9	2	2.1	2.2	2.3	2.4	2.5	2.6	2.7	≥2.8
$f(a)$	1	0.85	0.77	0.71	0.66	0.63	0.6	0.58	0.56	0.55	0.53	0.52	0.51	0.51	0.5

Subject to

$$g_1(\vec{x}) = W - W_s \geq 0,\ g_2(\vec{x}) = P_{\max} - P_o \geq 0,$$
$$g_3(\vec{x}) = \Delta T_{\max} - \Delta T \geq 0,\ g_4(\vec{x}) = h - h_{\min} \geq 0,$$
$$g_5(\vec{x}) = R - R_o \geq 0,\ g_6(\vec{x}) = 0.001 - \frac{\gamma}{gP_o}\left(\frac{Q}{2\pi Rh}\right) \geq 0,$$
$$g_7(\vec{x}) = 5000 - \frac{W}{\pi(R^2 - R_o^2)} \geq 0.$$

where $W = \frac{\pi P_o}{2}\frac{R^2 - R_o^2}{\ln(R/R_o)}$, $P_o = \frac{6\mu Q}{\pi h^3}\ln(R/R_o)$, $E_f = 9336Q\gamma C\Delta T$
$\Delta T = 2(10^P - 560)$, $P = \frac{\log_{10}\log 10(8.122\times 10^6\mu + 0.8) - C_1}{n}$, $h = \left(\frac{2\pi N}{60}\right)^2\frac{2\pi\mu}{E_f}$
$\left(\frac{R^4}{4} - \frac{R_o^4}{4}\right)$, $\gamma = 0.0307$, $C = 0.5$, $n = -3.55$, $C_1 = 10.04$, $W_s = 101\,000$, $P_{\max} = 1000$, $h_{\min} = 0.001$, $\Delta T_{\max} = 50$, $g = 386.4$, $N = 750$.

Variable range $1 \leq R \leq 16$, $1 \leq R_o \leq 16$, $10^{-6} \leq \mu \leq 16 \times 10^{-6}$, $1 \leq Q \leq 16$.

References

Ab Wahab, M.N., Nefti-Meziani, S., Atyabi, A., 2015. A comprehensive review of swarm optimization algorithms. PLoS One 10 (5), e012282.

Abedinia, O., Amjady, N., Ghasemi, A., 2014. A new metaheuristic algorithm based on shark smell optimization. Complexity 21 (5), 97–116.

Akay, B., Karaboga, D., 2012a. A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci 192, 120–142.

Akay, B., Karaboga, D., 2012b. Artificial bee colony algorithm for large-scale problems and engineering design optimization. J. Int. Manuf. 23 (4), 1001–1014.

Alba, E., Dorronsoro, B., 2005. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans. Evol. Comput. 9 (2), 126–142.

Askarzadeh, A., 2014. Bird mating optimizer: an optimization algorithm inspired by bird mating strategies. Commun. Nonlinear Sci. Numer. Simul. 19 (4), 1213–1228.

Askarzadeh, A., 2016. A novel metaheuristic method for solving constrained ngineering optimization problems: Crow search algorithm. Comput. Struct. 169, 1–12.

Bayraktar, Z., Komurcu, M., Bossard, J., Werner, D., 2013. The wind driven optimization technique and its application in electromagnetics. IEEE Trans. Antennas and Propagation 61 (5), 2745–2757.

Belegundu, A.D., 1982. A Study of Mathematical Programming Methods for Structural Optimization. Department of Civil and Environmental Engineering, University of Iowa, Iowa City, Iowa.

Beni, G., Wang, J., 1993. Swarm Intelligence in Cellular Robotic Systems, Robots and Biological Systems: Towards a New Bionics?. Springer, Berlin, Heidelberg, pp. 703–712.

Beyer, H.G., Schwefel, H.P., 2002. Evolution strategies-A comprehensive introduction. Nat. Comput. 1 (1), 3–52.

Birbil, Ş.I., Fang, S.C., 2003. An electromagnetism-like mechanism for global optimization. J. Glob. Optim. 25 (3), 263–282.

Borji, A., 2007. A new global optimization algorithm inspired by parliamentary political competitions. In: Gelbukh, A., Kuri Morales, Á.F. (Eds.), MICAI 2007: Advances in Artificial Intelligence. MICAI 2007. In: Lecture Notes in Computer Science, vol. 4827, Springer, Berlin, Heidelberg.

Chuang, C.L., Jiang, J.A., 2007. Integrated radiation optimization: inspired by the gravitational radiation in the curvature of space–time. In: Evolutionary Computation, CEC 2007, IEEE Congress on. IEEE, pp. 3157–3164.

Civicioglu, P., 2013. Backtracking search optimization algorithm for numerical optimization problems. Appl. Math. Comput. 219 (15), 8121–8144.

Coello, C.A.C., 2000a. Treating constraints as objectives for single-objective evolutionary optimization. Eng. Opt. A35 32 (3), 275–308.

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

Coello, C.A.C., Becerra, R.L., 2004. Efficient evolutionary optimization through the use of a cultural algorithm. Eng. Optim. 36, 219–236.

Coello, C.A.C., Montes, E.M., 2002. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv. Eng. Inf. 16, 193–203.

Cuevas, E., Cienfuegos, M., Zaldívar, D., Pérez-Cisneros, M., 2013. A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Syst. Appl. 40 (16), 6374–6384.

De Falco, I., Della Cioppa, A., Maisto, D., Scafuri, U., Tarantino, E., 2012. Biological invasion–inspired migration in distributed evolutionary algorithms. Inf. Sci. 207, 50–65.

Deb, K., Goyal, M., 1997. Optimizing engineering designs using a combined genetic search. In: Seventh International Conference on Genetic Algorithms. Ed. I. T. Back, pp. 512–528.

Dewar, H., Mous, P., Domeier, M., Muljadi, A., Pet, J., Whitty, J., 2008. Movements and site fidelity of the giant manta ray, manta birostris, in the komodo marine park, Indonesia. Mar. Biol. 155 (2), 121–133.

Digalakis, J., Margaritis, K., 2011. On benchmarking functions for genetic algorithms. Int. J. Comput. Math. 77, 481–506.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B 26 (1), 29–41.

Dos Santos Coelho, L., 2010. Gausslan quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Syst. Appl. 37 (2), 1676–1683.

Doğan, B., Ölmez, T., 2015. A new metaheuristic for numerical function optimization: Vortex search algorithm. Inf. Sci. 293, 125–145.

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

Flores, J.J., López, R., Barrera, J., 2011. Gravitational interactions optimization. In: International Conference on Learning and Intelligent Optimization. Springer, Berlin, Heidelberg, pp. 226–237.

Gajawada, S., 2016. Entrepreneur: Artificial human optimization. Trans. Mach. Learn. Artif. Intell. 4 (6), 64–70.

Gandomi, A.H., Alavi, A.H., 2012. Krill herd: a new bio-inspired optimization algorithm. Commun. Nonlinear Sci. Numer. Simul. 17 (12), 4831–4845.

Gaurav, D., Vijay, C., 2017. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Adv. Eng. Softw. 114, 48–70.

Geem, Z.W., Kim, J., Loganathan, G.V., 2001. A new heuristic optimization algorithm: harmony search. Trans. Simul. 76 (2), 60–68.

Genç, H.M., Eksin, I., Erol, O.K., 2010. Big bang-big crunch optimization algorithm hybridized with local directional moves and application to target motion analysis problem. In: Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on. IEEE, pp. 881–887.

Gene, H., George, H.B., 2014. Sharks: The Animal Answer Guide. Johns Hopkins University Press, Baltimore.

Gupta, S., Tiwari, R., Shivashankar, B.N., 2017. Multi-objective design optimization of rolling bearings using genetic algorithm. Mech. Mach. Theory 42, 1418–1443.

Hare, W., Nutini, J., Tesfamariam, S., 2013. A survey of non-gradient optimization methods in structural engineering. Adv. Eng. Softw. 59, 19–28.

He, S., Prempain, E., Wu, Q.H., 2004. An improved particle swarm optimizer for mechanical design optimization problems. Eng. Opt. 36 (5), 585–605.

He, Q., Wang, L., 2006. An effective co-evolutionary particle swarm optimization for engineering optimization problems. Eng. Appl. Artif. Intell. 20, 89–99.

He, Q., Wang, L., 2007. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl. Math. Comput. 186, 1407–1722.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, USA.

Huan, T.T., Kulkarni, A.J., Kanesan, J., Huang, C.J., Abraham, A., 2016. Ideology algorithm: a socio-inspired optimization methodology. Neural Comput. Appl. 1–32.

Huang, F.Z., Wang, L., He, Q., 2007. An effective co-evolutionary differential evolution for constrained optimization. Appl. Math. Comput. 186 (1), 340–356.

Javidy, B., Hatamlou, A., Mirjalili, S., 2015. Ions motion algorithm for solving optimization problems. Appl. Soft Comput. 32, 72–79.

Johnna, R., 2016. Ocean Animals: Who's Who in the Deep Blue. National Geographic Washington D.C.

Juste, K.A., Kita, H., Tanaka, E., Hasegawa, J., 1999. An evolutionary programming solution to the unit commitment problem. IEEE Trans. Power Syst. 14 (4), 1452–1459.

Kannan, B.K., Kramer, S.N., 1994. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J. Mech. Des. 116, 405–411.

Karaboga, D., Akay, B., 2009. A comparative study of artificial bee colony algorithm. Appl. Math. Comput. 214 (1), 108–132.

Kashan, A.H., 2009. League championship algorithm: a new algorithm for numerical function optimization. In: International Conference on Soft Computing and Pattern Recognition, SOCPAR09. IEEE, Singapore, pp. 43–48.

Kaur, R., Kumar, R., Bhondekar, A.P., Kapur, P., 2013. Human opinion dynamics: an inspiration to solve complex optimization problems. Sci. Rep. 3 (3008).

Kaveh, A., Bakhshpoori, T., 2016. Water evaporation optimization: a novel physically inspired optimization algorithm. Comput. Struct. 167, 69–85.

Kaveh, A., Dadras, A., 2017. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. Adv. Eng. Softw. 110, 69–84.

Kaveh, A., Farhoudi, N., 2013. A new optimization method: dolphin echolocation. Adv. Eng. Softw. 59, 53–70.

Kaveh, A., Talatahari, S., 2010. A novel heuristic optimization method: charged system search. Acta Mech. 213 (3), 267–289.
Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Vol. 194. pp. 2–8.
Kiran, M.S., 2015. TSA: Tree-seed algorithm for continuous optimization. Expert Syst. Appl. 42 (19), 6686–6698.
Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.
Krause, J., Cordeiro, J., Parpinelli, R.S., Lopes, H.S., 2013. A survey of swarm algorithms applied to discrete optimization problems. In: Swarm Intelligence and Bio-Inspired Computation: Theory and Applications. Elsevier Sci. Technol. Books, pp. 169–191.
Krihnanand, K.N., Ghose, D., 2009. Glowworm swarm optimization for simultaneous Capture of multiple local optima of multimodal functions. J. Swarm Intell. 2 (3), 87–124.
Kripka, M., Kripka, R.M.L., 2008. Big crunch optimization method. In: International Conference on Engineering Optimization, Brazil. pp. 1–5.
Kumar, M., Kulkarni, A.J., Satapathy, S.C., 2018. Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology. Future Gener. Comput. Syst. 81, 252–272.
Kuo, H.C., Lin, C.H., 2013. Cultural evolution algorithm for global optimizations and its applications. J. Appl. Res. Technol. 11 (4), 510–522.
Liang, J.J., Qu, B.Y., Suganthan, P.N., 2013. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. In: Zhengzhou China and Technical Report. Computational Intelligence Laboratory, Zhengzhou University, Nanyang Technological University, Singapore.
Liu, H., Cai, Z., Wang, Y., 2010. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl. Soft Comput. 10, 629–640.
Liu, B., Wang, L., Jin, Y.H., Tang, F., Huang, D.X., 2005. Improved particle swarm optimization combined with chaos. Chaos Solitons Fractals 25 (5), 1261–1271.
Lynn, N., Suganthan, P.N., 2015. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm Evol. Comput. 24, 11–24.
Mehrabian, A.R., Lucas, C., 2006. A novel numerical optimization algorithm inspired from weed colonization. Ecol. Inf. 1 (4), 355–366.
Meng, Z., Pan, J.S., 2016. Monkey king evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization. Knowl.-Based Syst. 97, 144–157.
Mezura-Montes, E., Coello, C.A.C., 2005. Useful infeasible solutions in engineering optimization with evolutionary algorithms. In: MICAI 2005. Lect. Notes Artif. Int., Vol. 3789. pp. 652–662.
Miller, M.H., Klimovich, C., 2016. Endangered Species Act Status Review Report: Giant Manta Ray (Manta birostris) and Reef Manta Ray (Manta alfredi). Draft Report to National Marine Fisheries Service, Office of Protected Resources, Silver Spring, MD.
Mirjalili, S., 2016. SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133.
Mirjalili, S.A., Hashim, S.Z.M., 2012. BMOA: binary magnetic optimization algorithm. Int. J. Mach. Learn. Comput. 2 (3), 204.
Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67.
Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.
Moghaddam, F.F., Moghaddam, R.F., Cheriet, M., 2012. Curved space optimization: a random search based on general relativity theory. arXiv preprint arXiv:12082214.
Mohamed, A.A.A., Mohamed, Y.S., El-Gaafary, A.A.M., Hemeida, A.M., 2017. Optimal power flow using moth swarm algorithm. Electr. Power Syst. Res. 142190–142206.
Montes, E., Coello, C.A.C., Reyes, J.V., 2016. Increasing successful offspring and diversity in differential evolution for engineering design. In: Proceedings of the Seventh International Conference on Adaptive Computing in Design and Manufacture. pp. 131–139, 2006b.
Moosavian, N., Roodsari, B.K., 2014. Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evol. Comput. 17, 14–24.
Moscato, P., Mendes, A., Berretta, R., 2007. Benchmarking a memetic algorithm for ordering microarray data. Biosystems 88 (1), 56–75.
Mucherino, A., Seref, O., 2007. Monkey search: a novel metaheuristic search for global optimization. In: AIP Conference Proceedings, Vol. 953(1). AIP, pp. 162–173.
Mühlenbein, H., Gorges-Schleuter, M., Krämer, O., 1988. Evolution algorithms in combinatorial optimization. Parallel Comput. 7 (1), 65–85.
Ngo, T.T., Sadollah, A., Kim, J.H., 2016. A cooperative particle swarm optimizer with stochastic movements for computationally expensive numerical optimization problems. J. Comput. Sci. 13, 68–82.
Oftadeh, R., Mahjoob, M.J., Shariatpanahi, M., 2010. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. Comput. Math. Appl. 60 (7), 2087–2098.
Osyczka, A., 2002. Evolutionary Algorithms for Single and Multicriteria Design Optimization: Studies in Fuzzyness and Soft Computing. PhysicaVerlag, Heidelberg.
Pan, W.T., 2012. A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl.-Based Syst. 266, 9–74.
Parsopoulos, K.E., Vrahatis, M.N., 2005. Unified particle swarm optimization for solving constrained engineering optimization problems. In: International Conference on Natural Computation. Springer, Berlin, Heidelberg, pp. 582–591.
Passino, K.M., 2002. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. 22 (3), 52–67.
Patel, V.K., Savsani, V.J., 2015. Heat transfer search (HTS): a novel optimization algorithm. Inf. Sci. 324, 217–246.
Punnathanam, V., Kotecha, P., 2016. Yin-yang-pair optimization: A novel lightweight optimization algorithm. Eng. Appl. Artif. Intell. 546, 2–79.
Rao, R.V., Savsani, V.J., Vakharia, D.P., 2011. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput.-Aided Des. 43 (3), 303–315.
Rao, R.V., Savsani, V.J., Vakharia, D.P., 2012. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf. Sci. 183 (1), 1–15.
Rao, B.R., Tiwari, R., 2007. Optimum design of rolling element bearings using genetic algorithms. Mech. Mach. Theory 42 (2), 233–250.
Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S., 2009. GSA: a gravitational search algorithm. Inf. Sci. 179 (13), 2232–2248.
Ray, T., Liew, K.M., 2003. Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Trans. Evol. Comput. 7 (4), 386–396.
Rebecca, S., Bobbie, K., 2015. Skates and Rays. Crabtree Pub Co.
Rocca, P., Oliveri, G., Massa, A., 2011. Differential evolution as applied to electromagnetics. IEEE Antennas Propag. Mag. 53 (1), 38–49.
Sacco, W.F., De Oliveira, C.R.E., 2005. A new stochastic optimization algorithm based on a particle collision metaheuristic. In: Proceedings of 6th WCSMO.
Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M., 2013. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. Appl. Soft Comput. 13 (5), 2592–2612.
Saremi, S., Mirjalili, S., Lewis, A., 2017. Grasshopper optimisation algorithm: Theory and application. Adv. Eng. Softw. 105, 30–47.
Satapathy, S., Naik, A., 2016. Social group optimization (SGO): a new population evolutionary optimization technique. Complex Intel. Syst. 2 (3), 173–203.
Shah-Hosseini, H., 2009. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspired Comput. 1 (1–2), 71–79.
Shah-Hosseini, H., 2011. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimization. Int. J. Comput. Sci. Eng. 6 (1–2), 132–140.
Shen, J., Li, Y., 2009. Light ray optimization and its parameter analysis. In: Computational Sciences and Optimization, CSO 2009, International Joint Conference on, Vol. 2. IEEE, pp. 918–922.
Siddall, J.N., 1982. Optimal Engineering Design. Marcel Dekker.
Simon, D., 2009. Biogeography-based optimization. IEEE Trans. Evol. Comput. 12 (6), 702–713.
Tamura, K., Yasuda, K., 2011. Primary study of spiral dynamics inspired optimization. IEEE Trans. Electr. Electron. Eng. 6 (S1), S98–S100.
Uymaz, S.A., Tezel, G., Yel, E., 2015. Artificial algae algorithm (AAA) for nonlinear global optimization. Appl. Soft Comput. 31, 153–171.
Wang, Y., Cai, Z., Zhou, Y., Fan, Z., 2009. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint handling technique. Struct. Multidiscipl. Optim. 37, 395–413.
Wang, L., Li, L.P., 2010. An effective differential evolution with level comparison for constrained engineering design. Struct. Multidiscipl. Optim. 41, 947–963.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Xu, Y., Cui, Z., Zeng, J., 2010. Social emotional optimization algorithm for nonlinear constrained optimization problems. In: Swarm, Evolutionary, and Memetic Computing, SEMCCO 2010. In: Lecture Notes in Computer Science, vol. 6466, Springer, Berlin, Heidelberg, pp. 583–590.
Yang, X.S., 2010. Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio-Inspired Comput. 2 (2), 78–84.
Yang, X.S., Deb, S., 2009. Cuckoo search via Lévy flights. In: Nature & Biologically Inspired Computing, NaBIC 2009, World Congress on. IEEE, pp. 210–214.
Yang, X.S., Hossein Gandomi, A., 2012. Bat algorithm: a novel approach for global engineering optimization. Eng. Comput. 29 (5), 464–483.
Zarand, G., Pazmandi, F., Pál, K.F., Zimányi, G.T., 2002. Using hysteresis for optimization. Phys. Rev. Lett. 89 (15), 150201.
Zhang, M., Luo, W., Wang, X., 2008. Differential evolution with dynamic stochastic selection for constrained optimization. Inf. Sci. 1783043–1783074.
Zhao, W., Wang, L., 2016. An effective bacterial foraging optimizer for global optimization. Inf. Sci. 329, 719–735.
Zhao, W., Wang, L., Zhang, Z., 2019a. A novel atom search optimization for dispersion coefficient estimation in groundwater. Future Gener. Comput. Syst. 91, 601–610.
Zhao, W., Wang, L., Zhang, Z., 2019b. Neural Comput. Appl. http://dx.doi.org/10.1007/s00521-019-04452-x.
Zhao, W., Wang, L., Zhang, Z., 2019c. Supply–demand-based optimization: a novel economics-inspired algorithm for global optimization. IEEE Access 7, 73182–73206.
Zheng, Y.J., 2015. Water wave optimization: a new nature-inspired metaheuristic. Comput. Oper. Res. 55, 1–11.
Zheng, M., Liu, G., Zhou, C., Liang, Y., Wang, Y., 2010. Gravitation field algorithm and its application in gene cluster. Algorithms Mol. Biol. 5 (1), 32.