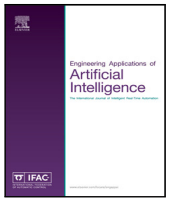
Engineering Applications of Artificial Intelligence 114 (2022) 105082

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems

Liying Wang [a], Qingjiao Cao [a], Zhenxing Zhang [b], Seyedali Mirjalili [c,d], Weiguo Zhao [a,*]

[a] School of Water Conservancy and Hydropower, Hebei University of Engineering, Handan, Hebei, 056038, China
[b] Prairie Research Institute, University of Illinois at Urbana-Champaign, Champaign, IL 61820, USA
[c] Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Fortitude Valley, Brisbane, 4006, QLD, Australia
[d] YFL (Yonsei Frontier Lab), Yonsei University, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
Artificial rabbits optimization
Nature-inspired algorithm
Meta-heuristic algorithm
Engineering problems
Fault diagnosis

ABSTRACT

In this paper, a new bio-inspired meta-heuristic algorithm, named artificial rabbits optimization (ARO), is proposed and tested comprehensively. The inspiration of the ARO algorithm is the survival strategies of rabbits in nature, including detour foraging and random hiding. The detour foraging strategy enforces a rabbit to eat the grass near other rabbits' nests, which can prevent its nest from being discovered by predators. The random hiding strategy enables a rabbit to randomly choose one burrow from its own burrows for hiding, which can decrease the possibility of being captured by its enemies. Besides, the energy shrink of rabbits will result in the transition from the detour foraging strategy to the random hiding strategy. This study mathematically models such survival strategies to develop a new optimizer. The effectiveness of ARO is tested by comparison with other well-known optimizers by solving a suite of 31 benchmark functions and five engineering problems. The results show that ARO generally outperforms the tested competitors for solving the benchmark functions and engineering problems. ARO is applied to the fault diagnosis of a rolling bearing, in which the back-propagation (BP) network optimized by ARO is developed. The case study results demonstrate the practicability of the ARO optimizer in solving challenging real-world problems. The source code of ARO is publicly available at https://seyedalimirjalili.com/aro and https://ww2.mathworks.cn/matlabcentral/fileexchange/110250-artificial-rabbits-optimization-aro.

1. Introduction

Over the past few decades, meta-heuristic algorithms have been increasingly popular in dealing with challenging optimization problems in all kinds of engineering fields. This is because such techniques are more inexpensive and efficient than conventional numerical approaches. The merits of meta-heuristics lie in many aspects. The first is their randomness, which can ensure success in avoiding local extrema and exploring the search space. The second is the black box concept, in which the input and output of considered problems are used without the need of gradient information. In addition, meta-heuristic algorithms are easy to implement and their mathematical models are simple. With these merits, meta-heuristics are especially suitable for tackling nonlinear, high-dimensional, and multimodal problems in literature (Barshandeh et al., 2020; Barshandeh and Haghzadeh, 2020; Boussaïd et al., 2013; Dhiman and Kaur, 2019; Drira et al., 2020; Liao et al., 2021; Siarry, 2016; Wu et al., 2015).

Most meta-heuristics come from natural inspiration, and they can be divided into five categories as shown in Fig. 1: evolution-based (EB), swarm-based (SB), physics/chemistry-based (PCB), human-based (HB) algorithms, and others. EBs model natural evolution such as natural selection and species migration (Shefaei et al., 2018). Genetic algorithm (GA), one of the most common EBs, simulates Darwinian natural selection (Holland, 1975). With its emergence, multiple other EBs are proposed, including differential evolution (DE) (Storn and Price, 1997), evolution strategy (ES) (Beyer and Schwefel, 2002), queen-bee evolution (QE) (Jung, 2003), just to name a few.

SBs are inspired from swarming behaviors in nature, which mimic self-organizations of social creatures (Ertenlice and Kalayci, 2018; Yi-Fei et al., 2022). One of the most well-known SBs is particle swarm optimization (PSO) (Eberhart and Kennedy, 1995) which mimics bird flocking behaviors. There are many other popular SBs. For example, firefly algorithm (FA) models the social behaviors of fireflies based on their flashing characteristics by which they interact (Yang, 2010); whale optimization algorithm (WOA) models foraging behaviors of humpback whales (Mirjalili and Lewis, 2016); artificial fish swarm optimization (AFSA) mimics fish behaviors of preying, swarming, and chasing (Li et al., 2002); salp swarm algorithm (SSA) mimics the

* Corresponding author.
E-mail address: zhaoweiguo@hebeu.edu.cn (W. Zhao).

https://doi.org/10.1016/j.engappai.2022.105082
Received 17 March 2022; Received in revised form 17 April 2022; Accepted 13 June 2022
Available online xxxx
0952-1976/

navigating and foraging mechanisms of salps (Mirjalili et al., 2017); wild horse optimizer (WHO) models the group living behaviors of wild horses in nature (Naruei and Keynia, 2021) artificial ecosystem-based optimization (AEO) models the life and foraging behaviors of living things in ecosystem (Zhao et al., 2020a); artificial bee colony (ABC) mimics the foraging behaviors of honey bees with three groups of bees (Karaboga and Akay, 2009); cuckoo search (CS) algorithm mimics the behaviors of brood parasitism of some cuckoo species (Yang and Deb, 2009); virus colony search (VCS) models swarm behaviors of diffusion and infection strategies of virus (Li et al., 2016); Harris hawks optimization (HHO) simulates the swarm behaviors and chasing style of Harris' hawks (Heidari et al., 2019); manta ray foraging optimization (MRFO) models three feeding strategies of chain, cyclone and somersault foragings of manta rays (Zhao et al., 2020b).

Remarkably, as an important part of SBs, some swarm-based optimization techniques integrated with applied mathematics and control theory also have achieved considerable success and produced important results in the field. Chen et al. (2020) proposed a novel evolved bat algorithm with fuzzy systems and applied it to a complex nonlinear system; Precup et al. (2021) designed a fuzzy controller tuning approach mixed with slime mould algorithm for servo systems; Yi-Fei et al. (2022) developed a fresh Fractional-order ant colony algorithm (FACA), using the fractional calculus to change the transition behaviors of the original IACA, with the simple one-step probability replaced with complex expressions; and Pozna et al. (2022) presented an effective hybrid optimizer by combining PSO and particle filter (PF), and validated the efficiency by tuning fuzzy control systems.

PCBs, as the third category of meta-heuristics, are generally inspired from some physical or chemical laws, including magnetic force, climate phenomena, gravity, river systems, etc (Abedinpourshotorban et al., 2016; Vahidi and Foroughi Nematolahi, 2019; Mohammadi et al., 2021). Gravitational search algorithm (GSA) is a popular PCB that is inspired from the Newton law of gravity. Depending on a certain gravitational force, agents attract each other according to the gravitational law (Rashedi et al., 2009). Artificial chemical reaction optimization algorithm (ACROA), as another representative of PCBs (Alatas, 2011), is originated from the chemical reaction in which reactants collide with each other to transfer one chemical substance to another. Some other popular PCBs are atom search optimization (ASO) (Zhao et al., 2019a), water cycle algorithm (WCA) (Eskandar et al., 2012), colliding bodies optimization (CBO) (Kaveh and Mahdavi, 2014), galaxy-based search algorithm (GbSA) (Shah-Hosseini, 2011), henry gas solubility optimization (HGSO) (Hashim et al., 2019), charged system search (CSS) (Kaveh and Talatahari, 2010a), lighting attachment procedure optimization (LAPO) (Nematollahi et al., 2017), water evaporation optimization (WEO) (Kaveh and Bakhshpoori, 2016), artificial physics optimization (APO) (Xie and Zeng, 2010), flow regime algorithm (FRA) (Tahani and Babayan, 2019), ions motion optimization (IMO) (Javidy et al., 2015), nuclear reaction optimization (NRO) (Wei et al., 2019), black hole optimization (BHO) (Hatamlou, 2013), equilibrium optimizer (EO) (Faramarzi et al., 2020), and intelligent water drop (IWD) (Hosseini, 2007).

As the fourth category of meta-heuristics, HBs have been developed by using various characteristics related to humans such as social, emotional, cultural, etc. There are some popular HBs in literature, including teaching–learning-based optimization (TLBO) (Rao et al., 2012), society and civilization algorithm (SCA) (Ray and Liew, 2003), focus group (FG) algorithm (Fattahi et al., 2018), poor and rich optimization (PRO) (Moosavi and Bardsiri, 2019), league championship algorithm (LCA) (Kashan, 2014), student psychology-based optimization (SPBO) (Das et al., 2020), tug of war optimization (TWO) (Kaveh, 2017), and human mental search (HMS) (Mousavirad and Ebrahimpour-Komleh, 2017).

Undoubtedly, there are also meta-heuristics that do not fit the above four categories due to different sources of inspiration such as mathematics, economics, music, etc. Some of these other meta-heuristics are stochastic paint optimizer (SPO) (Kaveh et al., 2020), supply–demand-based optimization (SDO) (Zhao et al., 2019b), five-elements cycle optimization algorithm (FECO) (Liu, 2017), backtracking optimization search algorithm (BSA) (Civicioglu, 2013), differential search algorithm (DSA) (Civicioglu, 2012), sine cosine algorithm (SCA) (Mirjalili, 2016), harmony search (HS) algorithm (Geem et al., 2001), and yin–yang-pair optimization (YYPO) (Punnathanam and Kotecha, 2016).

Although so many algorithms have been proposed, they have a common characteristic. The optimization process has two steps: exploration and exploitation (Lynn and Suganthan, 2015). In the exploration step, algorithms tend to search for better solutions far from the current peak in the search space, and this search process appears to be global and extensive. While in the exploitation steps, algorithms tend to improve the best solution found so far by searching the neighborhood of the solution, and this search process appears to be local and intensive. Obviously, exploitation and exploration conflict with each other when both work together for solving a problem. Generally, in the first half of iterations, exploration is dominant to search the entire variable space more globally and jump out to local optima. In the latter half of iterations, exploitation is dominant to search the local region around the best solution found so far. A successful algorithm should be able to maintain a right balance between exploration and exploitation steps, which can mitigate the problems of local extrema stagnation and immature convergence.

Regardless of numerous well-known optimization algorithms and their improved versions, the question is why new optimizers are still needed. First, according to the No Free Lunch Theorem of Optimization (Wolpert and Macready, 1997), existing optimizers display an equivalent performance on average if they are used to solve all possible optimization problems. This to say, the algorithm that performs the best for all the possible optimization tasks does not exist. Second, if the performance of an existed algorithm is improved by different operators regarding one class of problems, it is offset by the performance regarding other classes (Faramarzi et al., 2020). These motivate many experts to develop new optimizers. It is the main motivation of the present study.

This study develops a novel optimizer named artificial rabbits optimization (ARO), which is originated from the survival strategies of detour foraging and random hiding, after an overall set of 31 functions and five semi-real engineering problems are verified, this proposed optimizer is applied to the fault diagnosis of a rolling bearing. The verified and experiment results suggest the practicability of ARO. The main contributions of this study are highlighted as follows:

- A bio-inspired artificial rabbits optimization (ARO) is proposed and the survival strategies of rabbits are investigated thoroughly and modeled mathematically.
- There are three search strategies, including detour foraging strategy, random hiding strategy, and energy shrink strategy. The detour foraging strategy contributes to exploration; the random hiding strategy is dedicated to exploitation; and the energy shrink strategy enhances the balance between exploration and exploitation.
- ARO is implemented and tested on 31 benchmark test functions and 5 engineering design cases.
- The performance of ARO is compared with that of some classical and latest optimizers.
- The practicability of ARO is examined for the fault diagnosis of a rolling bearing.

The rest of this study is organized as follows. Section 2 reviews the related works. The inspiration and mathematical model of ARO are described in Section 3. The comparative results of ARO on a set of overall benchmarks and five semi-real engineering problems are provided in Section 4. Section 5 presents an application of ARO in the fault diagnosis of a rolling bearing. Section 6 summarizes this work with a discussion on future perspectives.

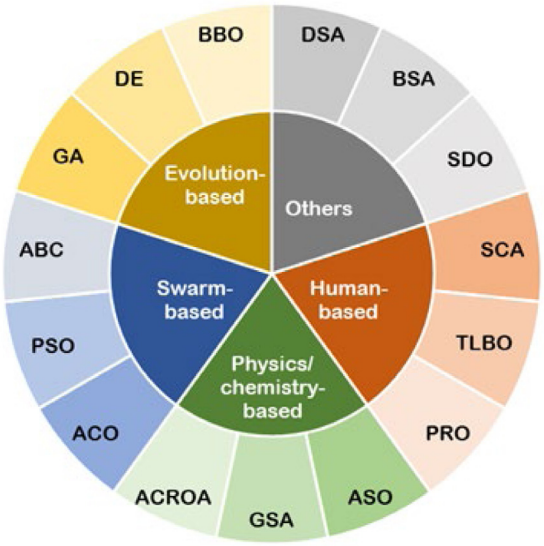


Fig. 1. Classification of meta-heuristic methods.

2. Related works

To understand the relevance of the proposed method, it is essential to investigate some latest well-known bio-inspired optimizers and some challenging applications.

With the development of computing power, numerous optimization algorithms have emerged and drawn much attention. Nevertheless, new optimizers are still being proposed to handle emerging complex optimization problems. Falcon optimization algorithm (FOA) (de Vasconcelos Segundo et al., 2019a) is a new bio-inspired optimization algorithm and it performs the global search by simulating hunt behaviors of falcons. The effectiveness of FOA is verified with application in heat exchangers shell-and-tube and plate-fin types. Black widow optimization (BWO) (Hayyolalam and Kazem, 2020) simulates the unique mating behaviors of black widow spiders, including procreation, cannibalism, and mutation. BWO is applied in solving three different challenging engineering design problems. Polar bear optimization (PBO) (Połap and Woźniak, 2017) is inspired by the way polar bears hunt to survive in harsh arctic conditions. This algorithm performs global and local searches by modeling two hunting phases of the polar bears and controls the size of individuals using birth and death mechanisms. Owl optimization algorithm (OOA) (de Vasconcelos Segundo et al., 2019b) originated from the decoy behaviors of the owls when any kind of danger is detected near the nests, and the algorithm is successfully applied in solving heat exchangers design problems. Cheetah based optimization algorithm (CBA) (Klein et al., 2018) is based on the social behaviors and hierarchy from the African cheetahs. The population in CBA is grouped into two parts according to genders: male and female, which adopt different search strategies. Coyote optimization algorithm (COA) (Pierezan and Coelho, 2018) simulates the social organization of the coyotes and the adaptability to its environment, establishing a balance between exploration and exploitation in the optimization process. To strengthen the search ability and reduce the control parameters of COA, an improved version of COA (MCOA) is proposed (Pierezan et al., 2021). In MCOA, the bell chaotic map is used for scatter and association probabilities tuning, and the updating parameter strategy based on social conditions is used to adjust the control parameters. MCOA enhances the ability to escape the local extrema and reduces the impact of parameter settings on the optimization performance. To enhance the exploratory ability of balancing exploration and exploitation of COA, a cultural COA (CCOA) (Pierezan et al., 2019) is developed. Some ideas from cultural algorithms are merged into CCOA (Reynolds, 1994), which is applied in a heavy-duty gas turbine operation. Zhao et al. (2020b) introduced the manta ray foraging optimization (MRFO) algorithm that is inspired by the intelligent foraging behaviors of manta rays in the sea. The optimizer performs the global search by mimicking three foraging behaviors. The current study may provide values other algorithms do not provide and it is complementary to the latest meta-heuristics for improving the performance of algorithms. Moreover, the current study also provides a possibility of learning from each other for solving real-world challenging problems.

With the development of society and technology, numerous complex and challenging optimization problems have emerged in different fields. As an illustration, one thorny question we are facing is the problem of spectrum resource shortage and high-speed access with the prosperity of 5G mobile communication network (Rasheed, 2022; Girjashankar and Upadhyaya, 2021; Israr et al., 2022). Through utilizing the spectrum information of network channel sharing to satisfy the quality of service (QoS) guarantee of users, the multi-agent system is embedded into an optimization method to determine the optimal allocation scheme by iteration (Ai et al., 2021). Another challenging optimization problem is how to manage training process and reduce communication cost in federated learning with the development of artificial intelligence (Hu et al., 2021; Rodríguez-Barroso et al., 2020; Chen et al., 2021). Połap and Woźniak (2021) proposed a new hybridization of optimization algorithms with federated learning for managing training process. This scheme uses the optimizer to manage the entire training process, and the best model is analyzed by minimizing the general model error. Park et al. (2021) proposed a scheme using an optimization algorithm to reduce communication cost in federal learning, which updates the global models via transmitting scores and gathering the weight of learned models. Other challenging optimization problems should also be treated with attention, including handwritten signature verification (Güler and Meghdadi, 2008), vehicles circuits optimization based on GPS/GSM information (Beldjilali et al., 2020), and reinforcement learning-based control (Zamfirache et al., 2022).

3. Artificial rabbits optimization (ARO)

The inspiration and idea behind the ARO algorithm are introduced. Then the mathematical models to simulate the survival strategies are investigated.

3.1. Basic idea

ARO is originated from the survival strategies of rabbits in nature. A brief explanation is given as follows.

Rabbits are herbivores that mainly feed on grass, forbs, and leafy weeds. Like any creatures else in our evolutionary history, rabbits have to evolve with survival (Tůmová et al., 2011). To prevent the nest from being found by predators, rabbits do not eat the grass near the holes, they always seek food away from their nests. Rabbits have a remarkably wide field of vision, and most of them are devoted to overhead scanning, so they can easily find food over a large area (Juan, 2017; Tynes, 2010). We will consider this detour foraging strategy as the exploration. This survival strategy of rabbits is widely spread as a popular Chinese idiom: “rabbits do not eat the grass near its own nest”.

Another survival strategy of rabbits is random hiding. Rabbits are good at making burrows for nests, to escape the predators or hunters’ track, a rabbit will dig many burrows around its nest and randomly choose one as a shelter from predators (Camp et al., 2014). Rabbits are foreleg short and long back legs, and they have strong muscles and tendons, which make them run faster. In addition, rabbits can also stop suddenly while running, turn sharply or run back to escape chase in a zigzag motion (Firestone and Warren, 2010). This survival strategy can easily confuse their enemies and help them escape tracking, which can effectively increase the survival probability of rabbits. We will consider

Fig. 2. A rabbit nest with several burrows [1].

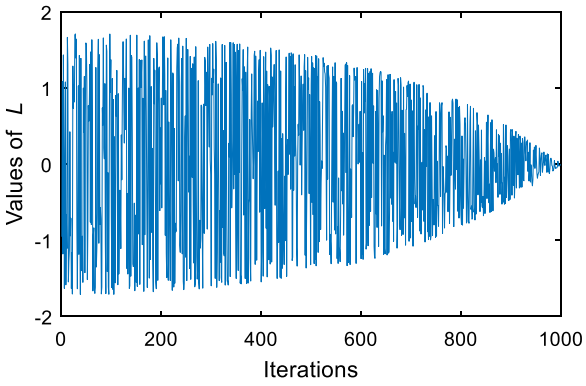


Fig. 3. Dynamic behavior of L during 1000 iterations.

this random hiding strategy as exploitation. Since rabbits are at the lower of the food chain and have so many predators, rabbits have to run fast to escape from danger, this will shrink their energy, so rabbits need to adaptively switch between detour foraging and random hiding according to the energy. Around 2200 years ago, Strategies of the Warring States (475-221 B.C.), a historic masterpiece in ancient China, recorded this survival strategy of rabbits: "the cunning rabbit has as many as three burrows, but it is just to stay alive". Fig. 2 illustrates a rabbit nest with several burrows.

3.2. Model and algorithm

ARO employs the foraging and hiding strategies of real rabbits, as well as their energy shrink leading to transiting between both strategies. The mathematical models are described and then the proposed ARO is outlined.

3.2.1. Detour foraging (exploration)

As mentioned above, when foraging, the rabbits seek far and neglect what lies close at hand. They only eat grass randomly in the other regions instead of in its own region, we refer to this foraging behavior as the detour foraging. In ARO, assume that each rabbit in the swarm has its own region with some grass and d burrows, and the rabbits always randomly visit the position of each other for foraging. Actually, when foraging, rabbits are likely to perturb around a food source to get enough food. Therefore, the detour foraging behavior of ARO signifies that each search individual tends to update its position towards the other search individual chosen randomly in the swarm and add a perturbation. The mathematical model of the detour foraging of rabbits is proposed:

$$\vec{v}_i(t+1) = \vec{x}_j(t) + R \cdot (\vec{x}_i(t) - \vec{x}_j(t)) + round(0.5 \cdot (0.05 + r_1)) \cdot n_1, \quad i, j = 1, \ldots, n \text{ and } j \neq i \tag{1}$$

$$R = L \cdot c \tag{2}$$

$$L = (e - e^{(\frac{t-1}{T})^2}) \cdot \sin(2\pi r_2) \tag{3}$$

$$c(k) = \begin{cases} 1 & if\ k == g(l) \\ 0 & else \end{cases} \quad k = 1, \ldots, d \text{ and } l = 1, \ldots, \lceil r_3 \cdot d \rceil \tag{4}$$

$$g = randperm(d) \tag{5}$$

$$n_1 \sim N(0, 1) \tag{6}$$

where $\vec{v}_i(t+1)$ is the candidate position of the ith rabbit at the time $t+1$, $\vec{x}_i(t)$ is the position of the ith rabbit at the time t, n is the size of a rabbit population, d is the dimension of the problem, T is the maximal number of iterations, $\lceil \cdot \rceil$ is the ceiling function, *round* indicates rounding to the nearest integer, *randperm* returns a random permutation of the integers from 1 to d, r_1, r_2 and r_3 are three random numbers in (0,1), L is the running length which represents the movement pace when performing the detour foraging, and n_1 is subject to the standard normal distribution.

In Eq. (1), the perturbation may assist ARO to avoid local extrema and perform global search. According to Eq. (3), the running length L can generate a longer step during the initial iterations. While this length can generate a shorter step during the later iterations. Fig. 3 gives the dynamic behavior of L, it can be seen that a longer step size can contribute to exploration while a shorter step size to exploitation. c is a mapping vector, which can help the algorithm randomly choose a random number of elements of search individuals to mutate in the foraging behavior. R represents the running operator which is employed to simulate the running characteristic of rabbits. Fig. 4 shows the trajectory of 300 steps of the running operator in 2-D and 3-D space. From Fig. 4, the search space is probed by a series of sudden turns with the random direction, and with the increase of steps this probe length gradually gets shorter.

Eq. (1) indicates that search individuals perform a random search for food according to the position of each other. This behavior allows a rabbit to move far away from its own region to the regions of the other rabbits. This special foraging behavior of rabbits who visit others' nests rather than their own nests significantly contributes to exploration and guarantees the global search capability of the ARO algorithm.

3.2.2. Random hiding (exploitation)

To escape from predators, a rabbit generally digs some different burrows around its nest for hiding. In ARO, at each iteration, a rabbit always produces d burrows around it along each dimension of the search space, and it always randomly chooses one from all burrows for hiding to reduce the probability of being preyed on. The following equation is given in this regard. The jth burrow of the ith rabbit is generated by:

$$\vec{b}_{i,j}(t) = \vec{x}_i(t) + H \cdot g \cdot \vec{x}_i(t), i = 1, \ldots, n \text{ and } j = 1, \ldots, d \tag{7}$$

$$H = \frac{T - t + 1}{T} \cdot r_4 \tag{8}$$

$$n_2 \sim N(0, 1) \tag{9}$$

$$g(k) = \begin{cases} 1 & if\ k == j \\ 0 & else \end{cases} \quad k = 1, \ldots, d \tag{10}$$

From Eq. (7), the d burrows are generated in the neighborhood of a rabbit along every dimension. H is the hiding parameter which

[1] https://www.freeimages.com/photo/the-rabbit-1363382. The authors acknowledge that the photo in Figure 2 is used under Content License Agreement with FreeImages.com and was taken by Amethyst, https://www.freeimages.com/photographer/Amethyst-31280.

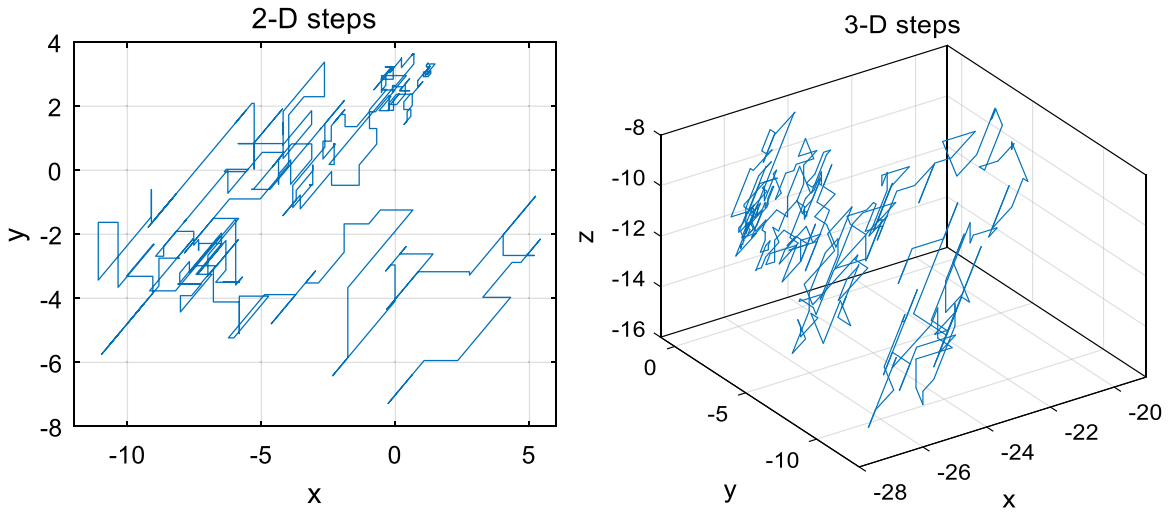


Fig. 4. Trajectory of 300 steps of the running operator.

is linearly decreased from 1 to $1/T$ with a random perturbation over the course of iterations (Mirjalili and Lewis, 2016). According to this parameter, initially, these burrows are generated in a bigger neighborhood of a rabbit. As the iterations increase, this neighborhood gets decreased, too.

As mentioned above, rabbits are often subjected to pursuit and attack from predators. To survive, rabbits need to find a safe place to hide. Therefore, they are declined to randomly select a burrow from their burrows for sheltering to avoid getting caught. To mathematically model this random hiding strategy, the equations are proposed:

$$\vec{v}_i(t+1) = \vec{x}_i(t) + R \cdot (r_4 \cdot \vec{b}_{i,r}(t) - \vec{x}_i(t)), i = 1, \dots, n \tag{11}$$

$$g_r(k) = \begin{cases} 1 & if \ \ k == \lceil r_5 \cdot d \rceil \\ 0 & else \end{cases} \qquad k = 1, \dots, d \tag{12}$$

$$\vec{b}_{i,r}(t) = \vec{x}_i(t) + H \cdot g_r \cdot \vec{x}_i(t) \tag{13}$$

where $\vec{b}_{i,r}$ represents a randomly selected burrow for hiding from its d burrows, and, r_4 and r_5 are two random numbers in (0,1). Based on Eq. (11), the ith search individual will try to update its position towards the randomly selected burrow from its d burrows.

After one of both detour foraging and random hiding is achieved, the position update of the ith rabbit is:

$$\vec{x}_i(t+1) = \begin{cases} \vec{x}_i(t) & f(\vec{x}_i(t)) \le f(\vec{v}_i(t+1)) \\ \vec{v}_i(\mathrm{t}+1) & f(\vec{x}_i(t)) > f(\vec{v}_i(t+1)) \end{cases} \tag{14}$$

This equation denotes that if the fitness of the candidate position of the ith rabbit is better than that of the current one, the rabbit will abandon the current position and stay at the candidate position generated by either Eq. (1) or Eq. (11).

3.2.3. Energy shrink (switch from exploration to exploitation)

In ARO, rabbits always tend to frequently perform detour foraging in the initial phase of iterations while frequently perform random hiding in the later phase of iterations. This search mechanism results from the energy of a rabbit, which will shrink gradually with the lapse of time. Therefore, an energy factor is designed to model the switch from exploration to exploitation. The energy factor in ARO is defined as follows:

$$A(t) = 4(1 - \frac{t}{T}) \ln \frac{1}{r} \tag{15}$$

where r is the random number in (0,1). Fig. 5 depicts the behavior curves of the energy factor. From Fig. 5, the energy factor $A(t)$ shows a decline trend towards zero during the iterations with the oscillation

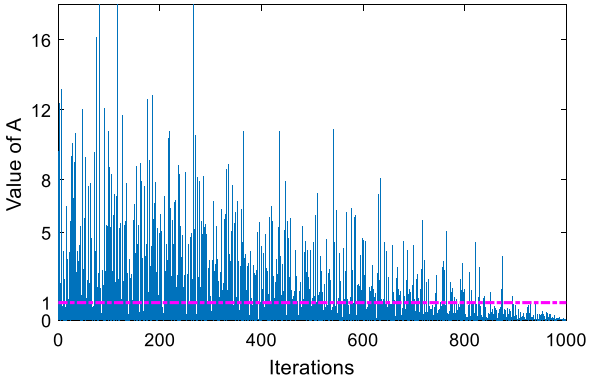


Fig. 5. Behavior of A during 1000 iterations.

amplitude. The big value of the energy factor represents that a rabbit has sufficient energy and physical strength for detour foraging. In contrast, the small value of the energy factor indicates that a rabbit is less physically active, so it needs random hiding. Therefore, in ARO, when the energy factor $A(t)$>1, a rabbit is prone to randomly explore the regions of different rabbits for foraging in the exploration phase, hence, the detour foraging happens; when the energy factor $A(t)\le 1$, a rabbit is inclined to randomly exploit its own burrows in the exploitation phase, thus, the random hiding occurs. Depending on the value of the energy factor A, ARO can switch between either the detour foraging or random hiding. That is to say, exploration occurs when $A(t)$>1, while exploitation happens when $A(t)\le 1$. The search mechanism depending on the energy factor is depicted in Fig. 6.

To investigate the influence of the energy factor on the search behavior of the algorithm, the probability of A>1 is calculated. Let $\theta = 2 \cdot (1 - \frac{t}{T})$, then $A(t) = 2 \cdot \ln \frac{1}{r} \cdot \theta$, the probability of A>0 is calculated by:

$$P\{A(t) > 1\} = \frac{\int_0^2 \int_0^{e^{-\frac{1}{2k}}} dr d\theta}{1 \cdot 2} = \frac{1}{4} \cdot \int_{-\infty}^{-\frac{1}{4}} \frac{e^t}{t} dt + e^{-\frac{1}{4}} \approx 0.5177 \tag{16}$$

So, the probability of detour foraging is about 0.5 in the iterative process. In other words, the ARO algorithm has almost the same amount of performing both detour foraging and random hiding in the iterative process, which contribute significantly to balancing exploration and exploitation. Fig. 7 shows the schematic diagram of probability calculation of the detour foraging.

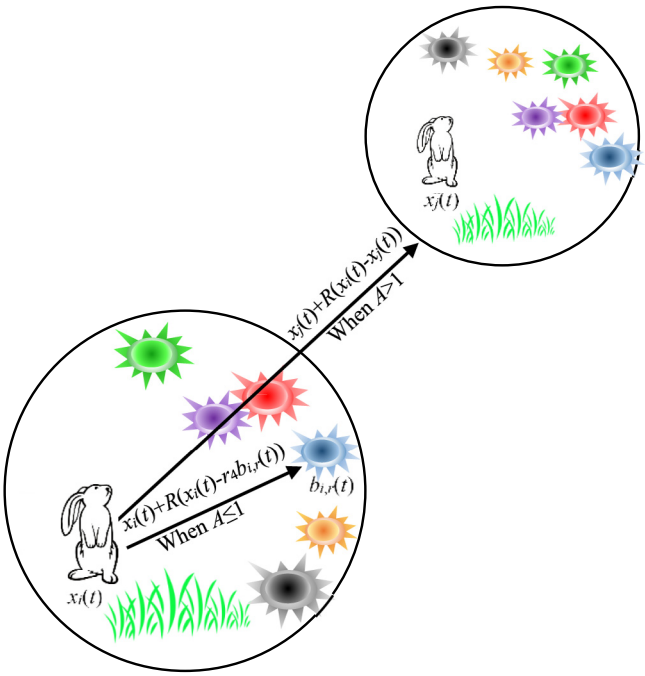


Fig. 6. Search mechanism based on the energy factor A.

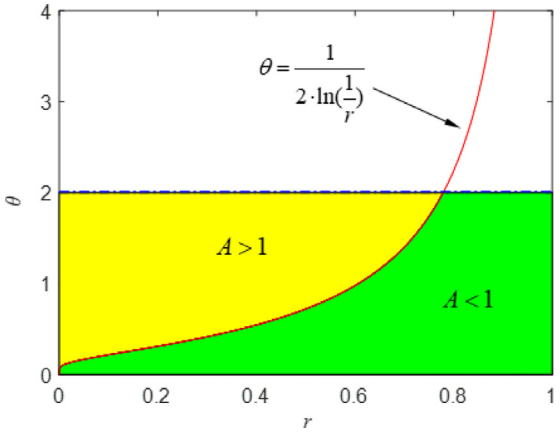


Fig. 7. Schematic diagram of probability calculation of detour foraging.

An important criterion for evaluating an optimization algorithm is whether it first emphasizes exploration and then exploitation, and these search behaviors have to exist during optimization (Mirjalili and Gandomi, 2017). In ARO, the search behaviors are controlled by the scope of the energy factor A, which gradually increases with the number of iterations with a random oscillator. On the one hand, the decreasing trend of A obliges the algorithm to frequently perform exploration in the early iterations while to frequently perform exploitation in the later iterations. It also assists the algorithm to switch slowly from global search to local search. On the other hand, the random oscillator does not change the decreasing trend of A which first contributes to exploration and then exploitation. Meanwhile, the energy factor A makes the algorithm has some exploration even in the final steps of iterations.

Taken together, the ARO algorithm generates a set of random population of artificial rabbits (candidate solutions) in the search space. At each iteration, a rabbit updates its position with respect to either a randomly chosen rabbit in the population or a randomly chosen one from its burrows. The energy factor A gets decreased with the increase of iterations, which can force each individual in the population to switch between the detour foraging behavior and random hiding behavior. All the updating and calculation are interactively achieved until the termination criterion is met, and then the best-so-far solution is returned. The pseudocode of ARO is described in Fig. 8 and the flowchart is provided in Fig. 9.

3.3. Computational complexity analysis

Computational complexity is very useful to evaluate the efficiency of an algorithm when solving optimization problems. Complexity is related to the number of individuals (n), the variable dimensionality of problems (d), and the maximum number of iterations (T). The overall computational complexity of ARO can be given as

$$\begin{aligned} &O(ARO) = O(problem\ definition) + O(initialization) \\ &+O(function\ evaluation) + O(Position\ updating\ in\ detour\ foraging) \\ &+O(Position\ updating\ in\ random\ hiding) \\ &= O(1+n+Tn+\frac{1}{2}Tnd+\frac{1}{2}Tnd) \cong O(Tnd+Tn+n) \end{aligned} \tag{17}$$

3.4. Swarm behaviors

To visually observe the search behaviors of the proposed mathematical models, a simulation of swarm behaviors is implemented. The swarm behaviors of 30 rabbits to search for the optimum of a unimodal and a multimodal function in a 3-D space are depicted in Figs. 10 and 11, respectively. The green balls represent 30 rabbits and the red points represent the global optimum. Initially, these 30 rabbits are randomly generated in the search space and explore the entire one. Along with the increasing of time t, all the rabbits gradually narrow the search area and exploit the neighborhood of the optimum. Finally, all the rabbits converge to the global optimum. Fig. 9 reveals a good exploitation ability of ARO and Fig. 11 demonstrates a good exploration ability. Obviously, this search feature can be extended to a n-D space.

3.5. Convergence analysis of ARO

The Markov model is used to analyze how ARO converges to the global optimum. Related definitions are provided and relevant theorems are proved (Zhang et al., 2013; Ning and Zhang, 2013; Zhao et al., 2020a).

Definition 1. The individual state of a rabbit is denoted by X, $X\in A$ and A is feasible solution space. The set of the state space of rabbits is expressed as $\boldsymbol{X} = \{X \mid X\in A\}$.

Definition 2. The swarm state of rabbits is denoted by $s = (x_1, x_2, \ldots, x_n)$, x_i is the state of the ith rabbit. The set of the state space of rabbit swarm is expressed as $\boldsymbol{S} = \{s = (x_1, x_2, \ldots, x_n) \mid x_i\in X, 1\leq i\leq n\}$.

Definition 3. For $\forall x_i\in s$, $\exists x_j\in s$, the state transition of a rabbit is denoted by $T_s(x_i) = x_j$.

Theorem 1. *In ARO, the probability of the state x_i of a rabbit transferred to x_j is expressed by:*

$$P(T_s(x_i)=x_j) = \begin{cases} P_d(T_s(x_i)=x_j) & Detour\ foraging \\ P_r(T_s(x_i)=x_j) & Random\ hiding \end{cases} \tag{18}$$

Randomly initialize a set of rabbits X_i (solutions) and evaluate their fitness Fit_i, and X_{best} is the best solution found so far.
While the stop criterion is not satisfied do
  For each individual X_i do
  Calculate the energy factor A using Eq. (15).
  If $A>1$
    Choose a rabbit randomly from other individuals.
    Calculate R using Eqs. (2)-(6).
    Perform detour foraging using Eq. (1).
    Calculate the fitness Fit_i.
    Update the position of the current individual using Eq. (14).
  Else
    Generate d burrows and randomly pick one as hiding using Eq. (13).
    Perform random hiding using Eq. (11).
    Calculate the fitness Fit_i.
    Update the position of the individual using Eq. (14).
  End If
  Update the best solution found so far X_{best}.
  End For
End While
Return X_{best}

Fig. 8. Pseudocode of ARO algorithm.

Proof. Based to Definition 3 and the geometric property of ARO, the transition probability of a rabbit is calculated:

$$P_p(T_s(x_i)=x_j)=\begin{cases}\dfrac{1}{2(e-e^{(\frac{t-1}{T})^2})|X_i-X_j|}p_1(x_i\to x_j)\\ \begin{cases}X_j\in[X_i-(e-e^{(\frac{t-1}{T})^2})(X_i-X_j),\\ \quad X_i+(e-e^{(\frac{t-1}{T})^2})(X_i-X_j)]\end{cases}\\ \dfrac{1}{2(e-e^{(\frac{t-1}{T})^2})|X_r-X_i|}p_1(x_i\to x_j)\\ \begin{cases}X_j\in[X_i-(e-e^{(\frac{t-1}{T})^2})(X_r-X_i),\\ \quad X_i+(e-e^{(\frac{t-1}{T})^2})(X_r-X_i)]\end{cases}\\ 0\qquad\text{otherwise}\end{cases} \tag{19}$$

$$p_1(x_i\to x_j)=\begin{cases}1 & f(x_j)<f(x_i)\\ 0 & f(x_j)\geq f(x_i)\end{cases} \tag{20}$$

Definition 4. In ARO, for $\forall s_i\in\boldsymbol{S}$ and $\forall s_j\in\boldsymbol{S}$, the transition probability of the swarm state s_i transferred to s_j is defined by

$$P(T_s(s_i)=s_j)=\prod_{k=1}^{m}P(T_s(X_{ik})=s_{jk}) \tag{21}$$

Theorem 2. *In ARO, the state sequence of the swarm {s(t): t≥0} is a finite homogeneous Markov chain.*

Proof. In the rabbits' state space $\boldsymbol{X}$, the position x of a rabbit is finite, thus the rabbit' state space $\boldsymbol{X}$ is finite. The swarm state space of rabbits $s(X_1, X_2, \ldots, X_m)$ is composed of m rabbits, and m is a finite positive integer. Thus, the swarm state space of rabbits $\boldsymbol{S}$ is also finite. Based on Definition 4, in the swarm sequence of rabbits $\{s(t): t\geq 0\}$, for $\forall s(t-1)\in\boldsymbol{S}$ and $\forall s(t)\in\boldsymbol{S}$, $P(T_s(s(t-1))=s(t))$ is determined by $P(T_s(X(t-1))=X(t))$. Based on Theorem 1, $P(T_s(X(t-1))=X(t))$ is only connected with the state $X(t-1)$, the running operator R, and other coefficients. $P(T_s(s(t-1))=s(t))$ is only connected with the individual state of rabbits at the $(t-1)$th time. So, the swarm state space is finite homogeneous Markov chain.

Theorem 3. *ARO can converge to the global optimum.*

Proof. ARO is a meta-heuristic algorithm, which satisfies Global Search Convergence Theorem (Solis and Wets, 1981). According to this Theorem, the probability that ARO cannot find the global optimum is 0 (Zhao et al., 2020a).

$$\prod_{k=0}^{\infty}P(1-u_k[B])=0 \tag{22}$$

where $u_k[B]$ is the probability measure satisfying the global convergence criterion (Solis and Wets, 1981). So, we can obtain:

$$\lim_{k\to\infty}P(x^k\in R_{\varepsilon,M})=1 \tag{23}$$

where $P(x^k\in R_{\varepsilon,M})$ is the probability the kth step, x^k is the solution produced by ARO in $R_{\varepsilon,M}$. According to the global convergence of heuristic algorithms, ARO can converge to the global optimum.

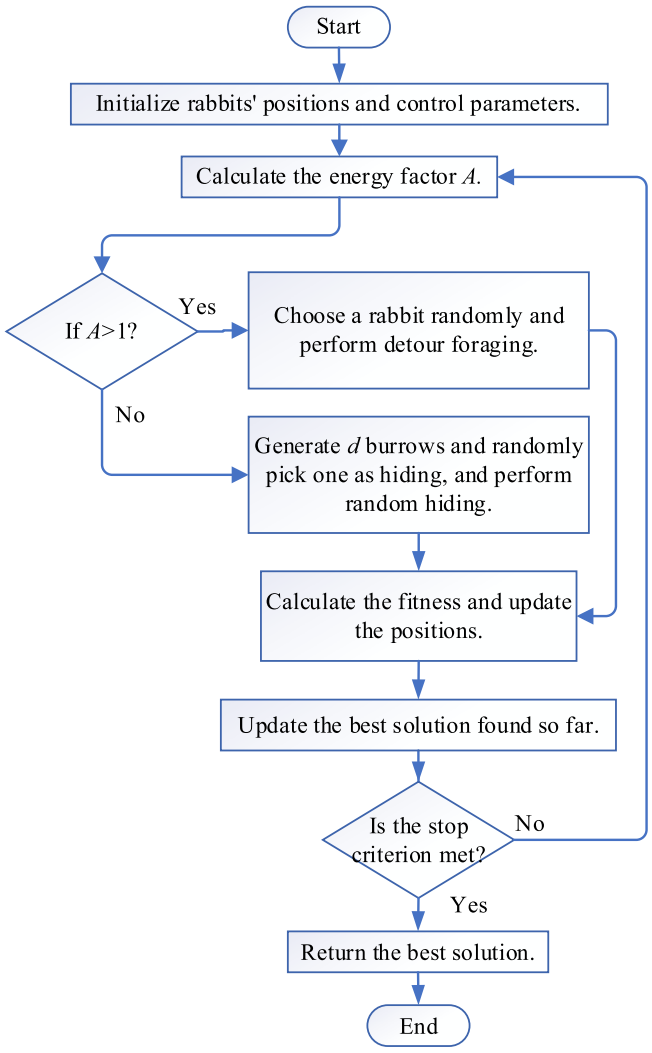


Fig. 9. Flowchart of ARO algorithm.

4. Results and discussions

4.1. Benchmark functions and compared algorithms

To measure the performance of the proposed ARO, a set of comprehensive benchmark functions are used in this experiment. This function set from literature (Duan et al., 2021; Zhao et al., 2022; Mirjalili et al., 2014) and the CEC 2014 special session (Liang et al., 2013) includes three types: unimodal functions (UFs), multimodal functions (MFs), and composition functions (CFs). The UFs (F1–F7) with one global optimum can test exploitation of optimization algorithms; the MFs (F8–F23) with plenty of local extrema can reveal exploration of optimizers; the CFs (F24–F31) can check the abilities of various optimizers to balance exploration and exploitation and avoid local extrema. Details of these benchmark functions can be found in Tables A.1–A.4 in Appendix.

Some well-known optimizers, including PSO, ASO, DE, CS, GSA, ABC, and TLBO, are used for comparisons with the ARO optimizer. These comparison results are based on the metrics of the best (Best), standard deviation (Std), and mean (Mean) of the best-so-far results given by each considered optimizer. The population size and maximum number of function evaluations (FEs) of each optimizer are set to 50 and 50,000, respectively. All the optimizers independently run 30 runs for each function and the results are based on the average performance of 30 runs. For ARO, no other parameters need to be adjusted except the population size and the maximum number of iterations. The parameters of other compared algorithms are based on recommendations in the literature (Storn and Price, 1997; Eberhart and Kennedy, 1995; Rashedi et al., 2009; Karaboga and Akay, 2009; Rao et al., 2012; Zhao et al., 2019a). The parameter settings of the other optimizers are given in Table 1.

Table 1
Parameter settings for each algorithm.

Algorithm	Parameter	Value
CS	Mutation probability	0.25
DE	Mutation factor Crossover rate	0.5 0.5
PSO	Cognitive coefficient Social coefficient	2 2
GSA	Gravitational constant Decreasing coefficient	100 20
ABC	Limit	$n{\cdot}d$
TLBO	Teaching factor	1,2
ASO	Depth weight Multiplier weight	50 0.2

4.2. Exploitation analysis

The solution accuracy and convergence of the UFs are very important since they have unique global optimum. Table 2 gives the comparisons provided by the investigated optimizers for the UFs. From Table 2, ARO can obtain the best mean results of best-so-far solutions for all UFs except for function F6. Meanwhile, ARO performs as well as ASO, CS, GSA, ABC, and TLBO for function F6. In total, the accuracy of the final solutions provided by ARO is the best for all the UFs. These competitive results demonstrate ARO has very efficient exploitation for the UFs. Fig. 12 depicts the comparisons of convergence curves for the UFs. The convergence of ARO tends to be accelerated during the entire iteration process. This means the search individuals can find the promising regions in the initial iterations and then speed up convergence. This behavior is evident in functions F1–F4 and F6. Therefore, ARO can expose a very good convergence rate when solving the UFs.

4.3. Exploration analysis

The quality of the final solutions is essential to algorithms since the MFs have multiple local extrema. Tables 3 and 4 list the comparisons for high-dimensional and fixed-dimensional MFs using various optimizers. Observing Table 3, the performance of ARO seems to be better than all other competitors for functions F8–F11. In addition, ARO can offer satisfactory results for functions F12 and F13. From Table 4, ARO can provide the best optimum for all functions except function F20. This is because of that the global search mechanism integrated into ARO can effectively enhance the level of exploration. Hence ARO is very competitive in exploiting the best solutions of the MFs, compared to its counterparts. The convergence curves of all the optimizers are shown in Figs. 13 and 14. From the figures, ARO maintains a stable convergence rate during the latter half of iterations. This is owing to the search individuals exploring the entire search space, and this obvious behavior can be observed for functions F8, F12, and F13–F23. The convergence curves in Figs. 13 and 14 can discover ARO outperforms the compared algorithms concerning the convergence rate in the majority of MFs.

4.4. Exploration–exploitation analysis

It is effective to avoid local extrema of functions by balancing exploration and exploitation. The CFs composed of various rotated and hybrid functions are very challenging, hence they are adapted to

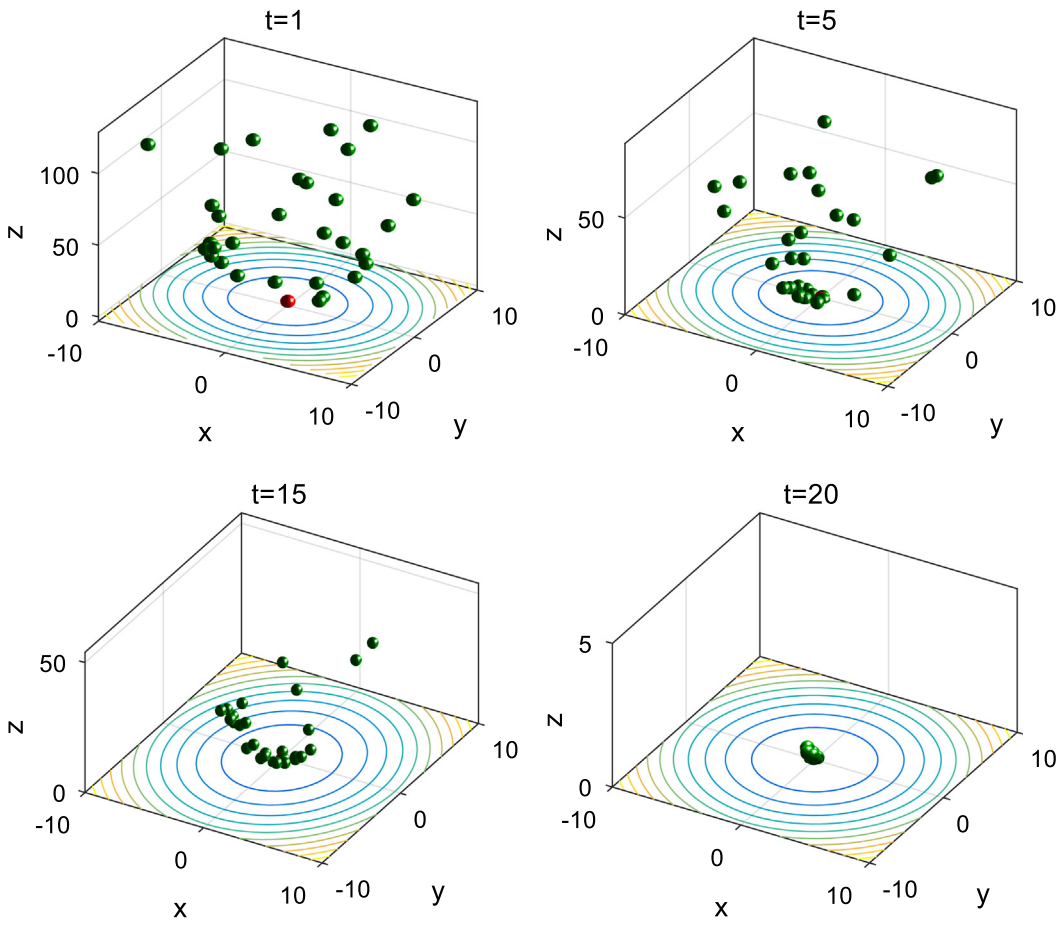


Fig. 10. Swarm behaviors of 30 individual (Sphere function).

Table 2
Comparisons of optimization results for UFs.

Function	Index	ARO	PSO	ASO	DE	CS	GSA	ABC	TLBO
	Mean	1.82E−124	2.15E−04	5.07E−22	3.64E−14	9.68E−03	2.19E−17	2.36E−03	3.64E−87
$F_1(x)$	Std	6.63E−124	2.25E−04	2.31E−22	6.06E−14	4.52E−03	6.38E−18	1.53E−03	1.67E−88
	Best	2.29E−142	2.99E−06	5.07E−22	1.26E−15	3.27E−03	1.15E−17	7.09E−04	1.53E−89
	Mean	2.68E−69	2.96E−04	1.24E−10	4.38E−08	1.40E+00	2.28E−08	2.32E−04	1.35E−43
$F_2(x)$	Std	1.21E−68	2.31E−04	1.82E−10	2.53E−08	5.61E−01	3.49E−09	1.53E−04	1.76E−43
	Best	1.44E−77	4.16E−05	1.24E−10	1.49E−08	6.31E−01	1.65E−08	8.11E−05	1.30E−89
	Mean	1.24E−95	2.84E+03	2.30E+02	5.69E+00	4.73E+02	2.22E+02	9.56E+03	5.65E−17
$F_3(x)$	Std	6.78E−94	1.34E+03	9.81E+01	3.91E+00	1.10E+02	7.07E+01	1.75E+03	2.28E−17
	Best	7.00E−115	1.14E+03	2.30E+02	9.26E−01	2.90E+02	8.53E+01	5.93E+03	4.02E−19
	Mean	9.92E−52	1.74E+01	2.34E−09	9.17E+00	3.25E+00	3.45E−09	2.45E+01	1.33E−35
$F_4(x)$	Std	2.96E−52	3.62E+00	2.25E−09	4.00E+00	8.55E−01	7.44E−10	2.28E+00	9.87E−36
	Best	1.84E−59	1.09E+01	2.34E−09	2.18E+00	1.60E+00	2.26E−09	1.98E+01	2.76E−36
	Mean	4.55E−03	9.47E+01	2.51E+01	3.00E+01	3.86E+01	2.67E+01	5.47E+02	2.14E+01
$F_5(x)$	Std	5.12E−03	7.90E+01	0.2338	1.76E+01	1.03E+01	2.67E+00	2.10E+02	1.11E+00
	Best	2.41E−04	7.62E+00	2.51E+01	4.01E+00	2.87E+01	2.57E+01	2.40E+02	1.76E+01
	Mean	0	1.33E−01	0	1.33E−01	0	0	0	0
$F_6(x)$	Std	0	4.34E−01	0	4.34E−01	0	0	0	0
	Best	0	0	0	0	0	0	0	0
	Mean	2.51E−04	5.64E−02	1.75E−02	2.15E−01	3.09E−02	1.91E−02	9.53E−02	5.62E−04
$F_7(x)$	Std	1.07E−04	2.03E−02	1.09E−02	7.24E−02	7.93E−03	6.87E−03	2.39E−02	1.72E−04
	Best	6.28E−05	1.92E−02	1.75E−02	1.16E−01	1.38E−02	7.73E−03	5.29E−02	1.47E−04

testing both exploration and exploitation simultaneously. Table 5 lists the comparisons of results provided by the algorithms for the CFs. As shown in Table 5, our method tends to outperform other optimizers for functions F24, F25, F26, F28 and F29. For function F30, ARO is superior to ASO, PSO, CS, GSA, and ABC. The convergence curves for the CFs are also depicted in Fig. 15, in which ARO shows a faster convergence speed over other algorithms on the majority of the CFs. In addition, the exploration–exploitation ratio is employed to measure the trade-off balance between the two searches (Hussain et al., 2018). Fig. 16 depicts the exploration–exploitation ratio provided by ARO when handling these composition functions. Inspecting Fig. 16, ARO achieves high exploration and low exploitation, and with the increase of iterations, the two search behaviors gradually shift. In the latter iterations, ARO performs exploitation higher than exploitation. ARO exhibits good abilities to balance exploration and exploitation and to escape local optima.

To further investigate the convergence performance of ARO in terms of exploratory and exploitative searches, the related experiment is

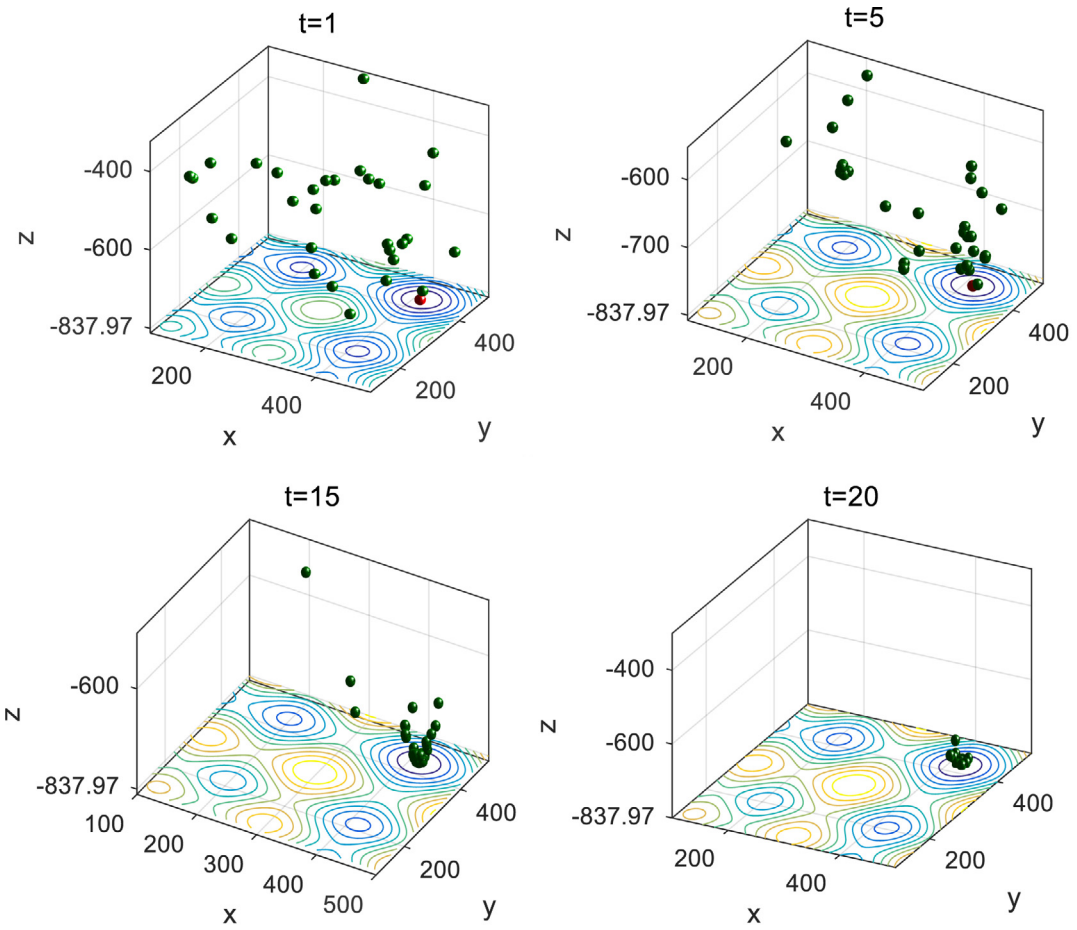


Fig. 11. Swarm behaviors of 30 individuals (Schwefel function).

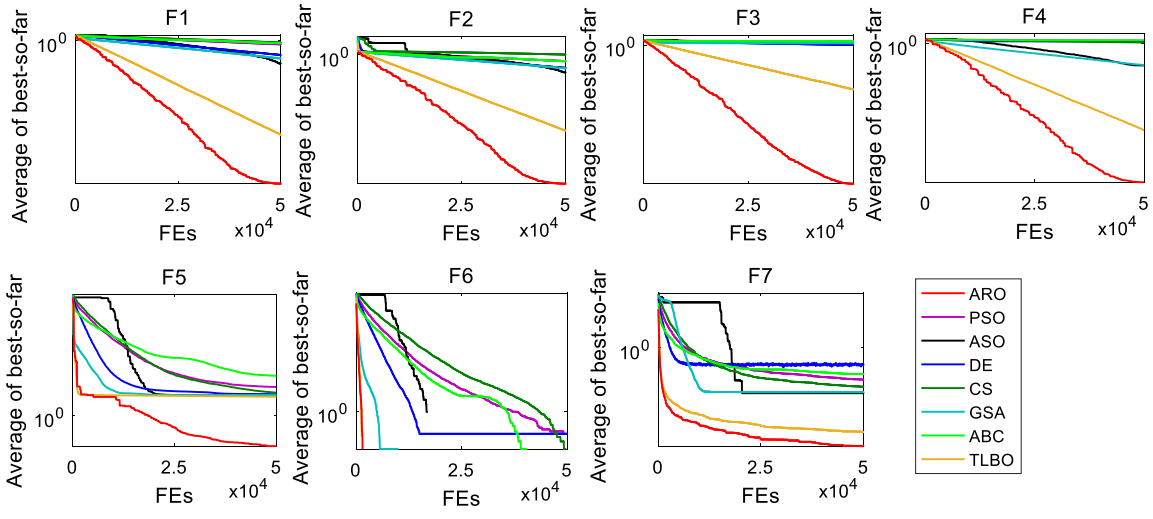


Fig. 12. Comparisons of convergence curves for UFs.

done on some test functions with two dimensions and a group of four rabbits are used over 100 iterations. Four metrics are used in the two-dimensional search space (Mirjalili, 2015). These four metrics are search history, trajectories of four rabbits in their two-dimensional space, average fitness of four rabbits, and convergence rate. The graphical results of four metrics are depicted in Fig. 17. The first metric is the first column of Fig. 17, which shows rabbits' position history over 100 iterations. The sampling points follow a similar distribution pattern on these functions: the sampling points in the area far from the global optimum are sparsely distributed, and the sampling points in the area close to the global optimum are densely distributed. This means that ARO is inclined to extensively explore the promising regions and intensively exploit the neighbor of the global optimum. Another metric is the second and third columns of Fig. 17, in which the maps of the trajectories with four different colors monitor how the first and second variables of the four rabbits vary during the iteration process.

Table 3
Comparisons of results for high-dimensional MFs.

Function	Index	ARO	PSO	ASO	DE	CS	GSA	ABC	TLBO
	Mean	−11209.6764	−5139.3732	−8424.05487	−5310.2672	−8691.8135	−2638.9144	−5123.7760	−7857.2640
$F_8(x)$	Std	462.3670	577.6833775	631.7281	661.6603	235.2491	435.1401613	460.6752	688.3056
	Best	−11996.76693	−6951.1321	−8424.055	−7387.3949	−9032.4037	−4189.0512	−6.53E+03	−9174.0450
	Mean	0	30.8756	25.6317	164.9567	83.2324	15.8530	157.8279	12.3234
$F_9(x)$	Std	0	8.8839	6.0892	17.5115	13.0994	3.8484	21.3024	5.8526
	Best	0	1.59E+01	1.39E+01	1.28E+02	5.89E+01	7.96E+00	1.11E+02	3.63E−01
	Mean	8.88E−16	7.60E−03	1.43E−11	5.41E−08	4.1525	3.44E−09	5.33E−02	6.34E−15
$F_{10}(x)$	Std	0	8.23E−03	2.82E−11	2.62E−08	1.4865	5.59E−10	4.05E−02	1.80E−15
	Best	8.88E−16	9.45E−04	1.43E−11	1.81E−08	2.02E+00	2.66E−09	1.24E−02	4.44E−15
	Mean	0	1.48E−02	0	2.05E−03	9.63E−02	4.2642	1.63E−01	0
$F_{11}(x)$	Std	0	1.36E−02	0	3.89E−03	4.18E−02	1.5875	1.15E−01	0
	Best	0	0.0001	0	7.99E−15	0.0261	1.8923	8.28E−03	0
	Mean	4.84E−08	0.3758	7.08E−24	6.91E−03	1.10E+00	3.40E−02	15.0140	3.46E−03
$F_{12}(x)$	Std	4.75E−08	7.26E−01	5.91E−24	2.63E−02	3.17E−01	5.36E−02	4.6712	1.89E−02
	Best	5.53E−09	1.44E−04	7.08E−24	1.33E−16	5.65E−01	5.16E−20	7.76E+00	2.82E−23
	Mean	3.67E−05	0.1909	4.76E−23	5.31E−02	1.34E−05	2.04E−18	36.2589	1.53E−02
$F_{13}(x)$	Std	4.09E−04	3.88E−01	6.03E−23	2.89E−01	6.31E−02	5.13E−19	18.1859	2.59E−02
	Best	1.04E−08	2.75E−03	4.76E−23	3.90E−15	6.29E−02	1.33E−18	1.29E+01	9.62E−22

Table 4
Comparisons of results for low-dimensional MFs.

Function	Index	ARO	PSO	ASO	DE	CS	GSA	ABC	TLBO
	Mean	0.998004	0.998004	0.998004	0.998004	0.998004	3.472815	0.9980054	0.998004
$F_{14}(x)$	Std	2.93E−15	2.51E−13	9.22E−12	4.26E−14	5.12E−13	2.529209	6.92E−06	0
	Best	0.9980038	0.998003838	0.998004	0.998004	0.998004	0.998004	0.99800384	0.9980038
	Mean	0.0003075	0.000366649	0.000802	0.0003075	0.0003075	0.002358	0.00050967	0.0003075
$F_{15}(x)$	Std	9.22E−18	0.000190754	1.65E−04	1.42E−19	3.92E−08	0.001143	4.99E−05	2.28E−19
	Best	0.0003075	0.000307486	5.42E−04	0.000307	0.000307	0.000991	0.00040355	0.0003075
	Mean	−1.031628	−1.031628	−1.03163	−1.03163	−1.03163	−1.03163	-1.031628	−1.031628
$F_{16}(x)$	Std	6.71E−16	6.78E−16	6.72E−16	6.78E−16	6.78E−16	5.53E−16	6.65E−16	6.78E−16
	Best	−1.031628	−1.03162845	−1.03163	−1.03163	−1.03163	−1.03163	−1.0316285	−1.031628
	Mean	0.3978874	0.397887358	0.397887	0.397887	0.397887	0.397887	0.3978874	0.3978874
$F_{17}(x)$	Std	0	0	0	0	0	0	6.75E−10	0
	Best	0.3978874	0.397887358	0.397887	0.397887	0.397887	0.397887	0.3978874	0.3978874
	Mean	3	3	3	3	3	3	3	3
$F_{18}(x)$	Std	1.99E−15	2.19E−15	2.21E−15	2.03E−15	1.88E−15	1.69E−15	2.86E−11	1.27E−15
	Best	3	3	3	3	3	3	3	3
	Mean	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278
$F_{19}(x)$	Std	2.71E−15	2.71E−15	0	2.71E−15	2.71E−15	2.45E−15	1.67E−15	2.71E−15
	Best	−3.862782	−3.86278215	−3.86278	−3.86278	−3.86278	−3.86278	−3.8627821	−3.862782
	Mean	−3.301995	−3.25407213	−3.322	−3.28633	−3.322	−3.322	−3.321995	−3.321995
$F_{20}(x)$	Std	4.34E−3	0.060415416	2.71E−15	0.055415	1.92E−13	1.36E−15	1.53E−15	1.37E−15
	Best	−3.321995	−3.32199517	−3.322	−3.322	−3.322	−3.322	−3.3219952	−3.321995
	Mean	−10.1532	−6.13117176	−7.6180	−9.98479	−10.1532	−6.97588	−10.110816	−10.1532
$F_{21}(x)$	Std	7.12E−15	2.835886928	2.7803	0.922443	3.80E−14	3.548979	0.16828352	6.96E−15
	Best	−10.1532	−10.1531997	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532
	Mean	−10.4029	−8.21793419	−10.4029	−10.4029	−10.4029	−10.4029	−10.4029	−10.4029
$F_{22}(x)$	Std	9.90E−16	2.978914237	2.38E−15	1.75E−15	2.80E−14	0	3.53E−14	1.36E−15
	Best	−10.40294	−10.4029406	−10.4029	−10.4029	−10.4029	−10.4029	−10.402941	−10.40294
	Mean	−10.5364	−7.71692326	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364
$F_{23}(x)$	Std	1.81E−15	3.250563289	1.35E−15	1.81E−15	4.20E−12	1.68E−15	8.09E−14	1.81E−15
	Best	−10.53641	−10.5364098	−10.5364	−10.5364	−10.5364	−10.5364	−10.53641	−10.53641

These trajectories share a common characteristic that the values of rabbits show an abrupt and large charge in the two-dimensional search space in the initial iterations. Nevertheless, this charge is weakened gradually in the subsequent iterations. This behavior can ensure that the swarm-based optimizer gradually converges towards a certain point in the search space (Van Den Bergh and Engelbrecht, 2006). This large and sudden change indicates exploration, while this small and slight change exploitation. The third metric is the fourth column of Fig. 17 which shows the mean objective values of four rabbits in each iteration. Obviously, according the curves, the decrease of the average fitness indicates that the accuracy of candidate solutions is improved. The four metric is the last column of Fig. 17 which gives the fitness of the best rabbit at each iteration. This continuous decrease in the fitness indicates the gradual convergence of ARO. These four metrics suggest ARO is very successful in improving the quality of candidate solutions for different problems.

4.5. Non-parametric statistical analysis

To better assess the performance of ARO, we conduct the Wilcoxon Signed-Rank Test (WSRT) on this experiment (Derrac et al., 2011). The WSRT can determine whether a certain statistical significance exists between the results of our method and other algorithms. The WSRT is conducted at 95% significance level. Tables 6 and 7 give the test results of the WSRT on 31 functions in 30 runs. '+' signifies that ARO has a 95% significance level with respect to the compared optimizers and '-' vice versa. '=' signifies that ARO has no significant difference between the compared optimizers. Table 8 gives the statistical results of WSRT between ARO and other algorithms. Based on Table 8, ARO displays a significant improvement over all other competitors for the majority of functions.

To test the ranks of our method and other algorithms, the Friedman test (Friedman, 1937), another popular non-parametric test, is conducted on this experiment study as well. The Friedman is also a

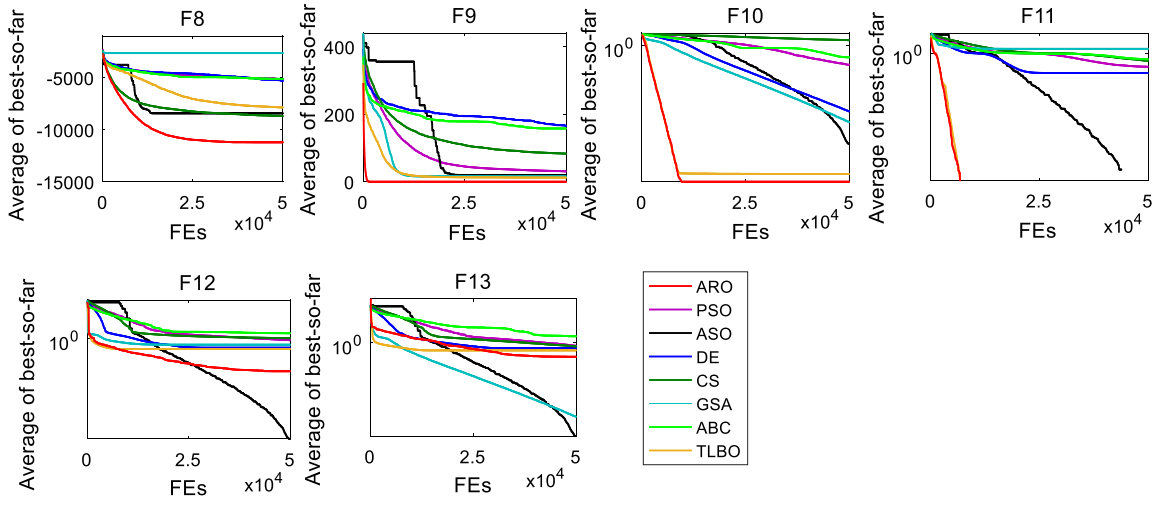


Fig. 13. Comparisons of convergence curves for high-dimensional MFs.

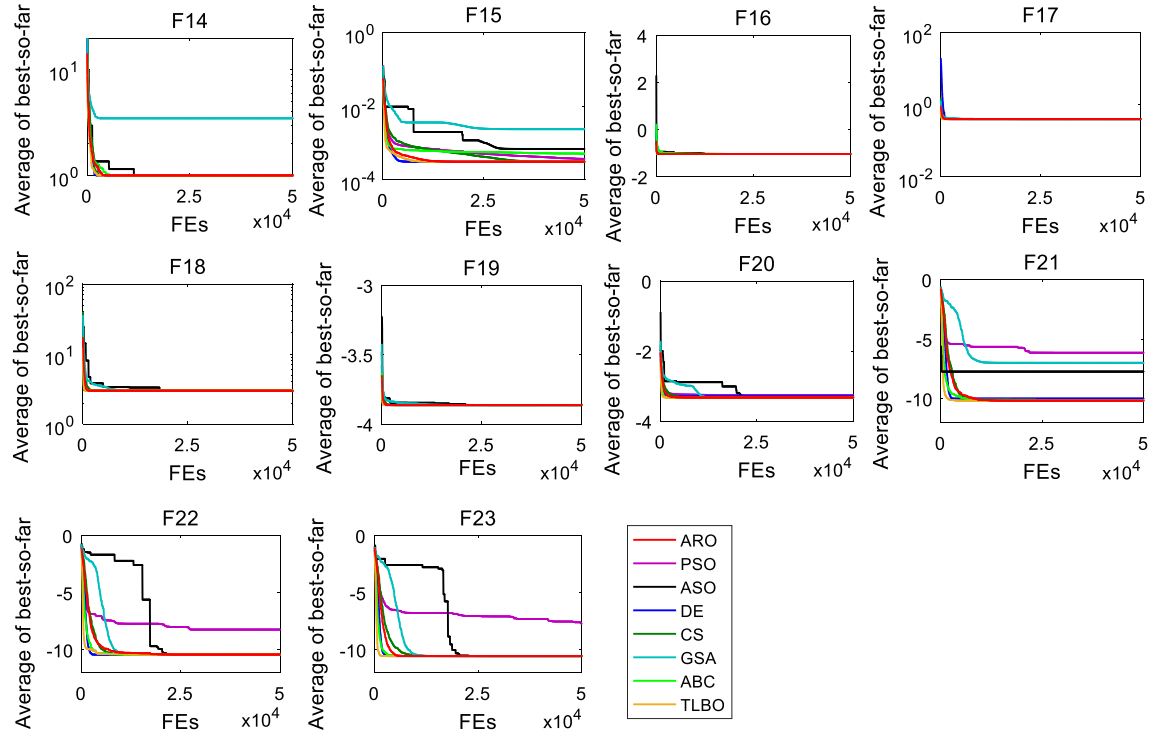


Fig. 14. Comparisons of convergence curves for fixed-dimensional MFs.

multiple comparisons test which can find significant differences among the behaviors of multiple optimizers, the ranks of the Friedman test are conducted on 31 functions for all the used algorithms. The ranks among all compared algorithms for each function are depicted in Fig. 18 using the radar chart and the mean of these ranks is shown in Fig. 19. Inspecting Fig. 19, ARO has the least average rank value, which indicates it ranks the first among all algorithms, highlighting ARO as the best performing optimizer of the comparison for the Friedman. This result proves once more that our method possesses the ability to find the global optimum for different problems efficiently.

4.6. Scalability analysis

A scalability assessment is employed to analyze the dimensionality effect on the problems using ARO. It can show how an optimizer maintains a stable searching advantage from lower dimensional tasks to higher dimensional tasks. In this test, ARO is used to solve the scalable unimodal and multimodal functions with the dimensionality varying from 50 to 500 with the step length 25. The scalability results of ARO versus other optimizers are depicted in Fig. 20. Based on the figure, ARO can exhibit superior results in all dimensions and its searching advantage is consistent when tackling the tasks with different dimensions. With the increase of the dimensionality, the optimization ability of ARO degrades less than other optimizers on the majority of the functions. This reveals that the effect of dimensionality on the solution quality is weaker for ARO than for other optimizers. Table 9 gives the results of the optimizers over 30 runs and 50,000 FEs for 500 dimensions of functions F1–F13. From Table 9, the results from ARO are obviously better than those from other optimizers and the best results are provided for 84.6% of the functions with 500 dimensions (i.e., F1–F5, F7–F8, and F10–F13). For functions F6 and F9, ARO and TLBO obtain the similar results, which are better than those of other

Table 5
Comparisons of results for composite functions.

Function	Index	ARO	PSO	ASO	DE	CS	GSA	ABC	TLBO
	Mean	2507.769	2.65E+03	2616.997	2615.244	2615.252	2534.215	2617.354	2615.244
$F_{24}(x)$	Std	29.265	11.87371047	1.320	6.77E−06	0.002609	63.424	4.58E−01	4.53E−09
	Best	2500	2647.739	2615.523	2615.244	2615.248	2500	2616.148	2615.244
	Mean	2600	2632.222	2624.690	2625.145	2634.032	2603.002	2634.199	2600.017
$F_{25}(x)$	Std	3.31E−07	7.351	1.121	2.570	3.393	6.041	2.031	2.28E−03
	Best	2600	2643.866	2621.911	2622.076	2633.281	2604.789	2630.805	2600.012
	Mean	2700	2722.703	2710.863	2703.768	2711.116	2701.248	2726.634	2700
$F_{26}(x)$	Std	0	7.350	1.303	0.444338	1.223	2.490	2.611	2.67E−13
	Best	2700	2722.969	2707.452	2702.869	2711.993	2700	2722.497	2700
	Mean	2743.5606	2704.985	2700.221	2700.379	2700.354	2793.516	2700.605	2736.962
$F_{27}(x)$	Std	50.198654	18.073	6.29E−02	6.85E−02	0.04997	16.973	0.09983139	48.799
	Best	2700.1861	2700.967	2700.100	2700.231	2700.389	2800.063	2700.352	2700.258
	Mean	3019.338	3384.102	3105.825	3036.477	3131.577	4372.258	3349.419	3221.784
$F_{28}(x)$	Std	166.06373	208.426	5.404	46.874	7.053	411.416	51.490	130.036
	Best	2900	3617.868	3101.803	3000.137	3131.861	4386.998	3242.390	3101.747
	Mean	3228.533	7047.004	4819.540	3655.013	3859.134	5169.593	4143.275	3959.748
$F_{29}(x)$	Std	156.28161	698.645	920.434	28.052	47.561	884.075	77.832	170.778
	Best	3000	7010.488	3100	3603.150	3859.627	4816.024	4046.280	3663.603
	Mean	4027.8285	1.39E+07	7.15+03	3894.941	7689.303	2.02E+06	1.55E+05	4127.043
$F_{30}(x)$	Std	802.71021	1.99E+07	1.15E+03	168.866	2140.874	1.11E+07	67244.425	319.221
	Best	3100	4756.596	49913.427	3510.029	9725.883	3100.154	6.54E+04	3563.555
	Mean	7952.476	1.49E+05	1.25E+04	4222.410	7833.551	1.95E+05	2.62E+04	5429.482
$F_{31}(x)$	Std	2113.2301	1.74E+05	7952.675	417.004	1182.403	8.83E+04	5137.156	689.708
	Best	3200	1.16E+05	6376.465	3571.536	8766.775	2.03E+05	1.70E+04	4208.880

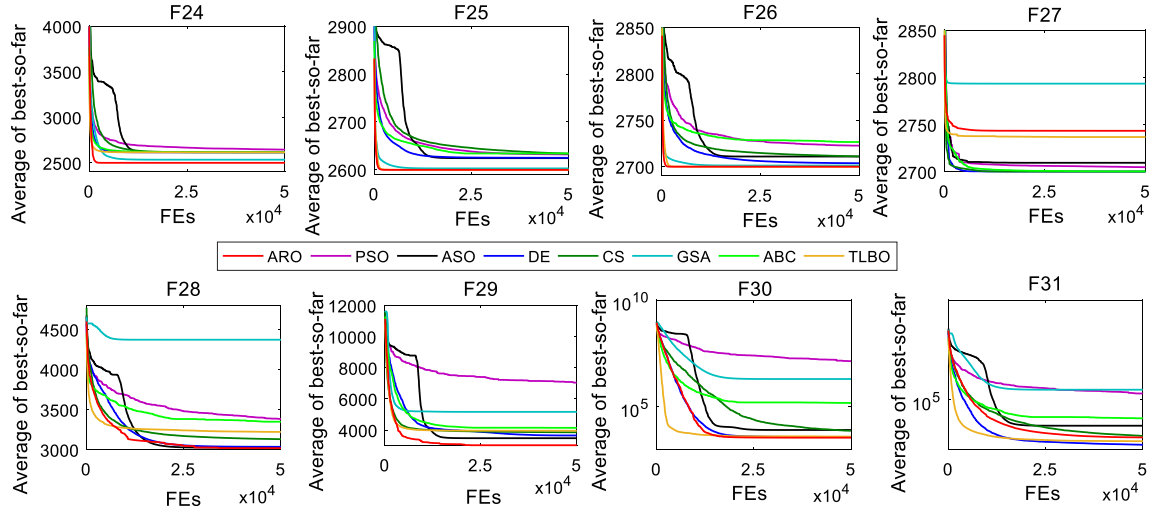


Fig. 15. Comparisons of convergence curves for composite functions.

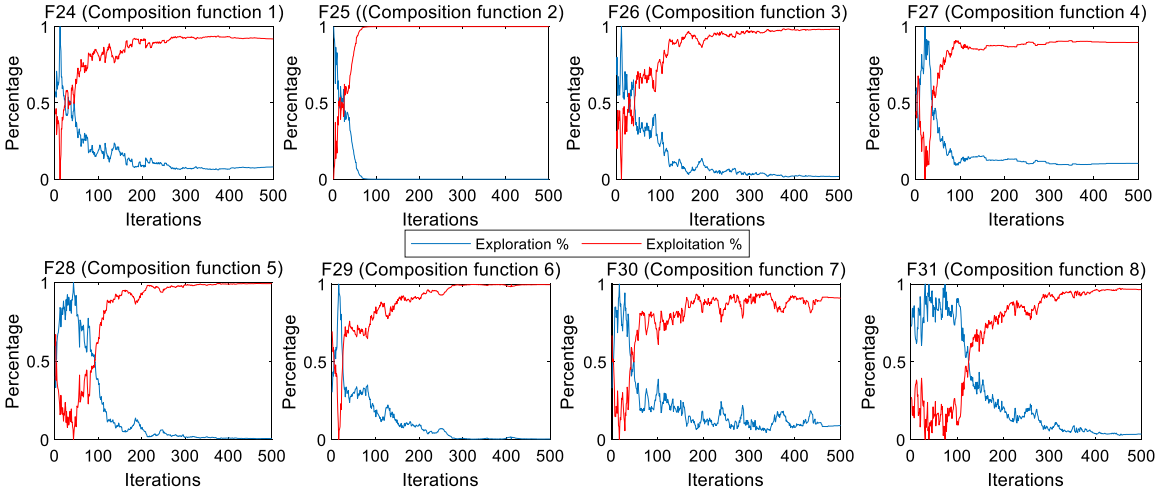


Fig. 16. Exploration and exploitation phases for ARO on composition functions.

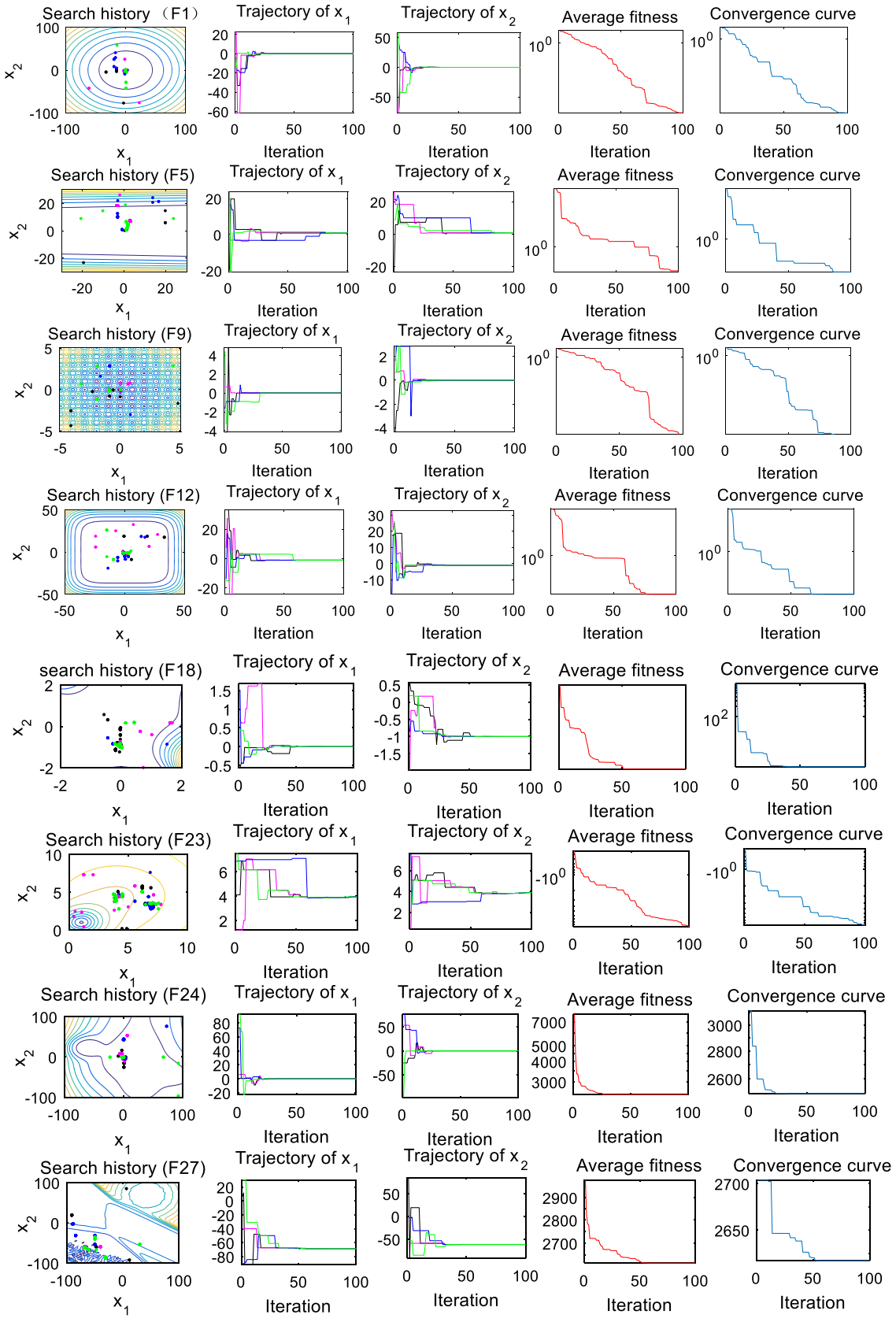


Fig. 17. Search history, trajectory in two dimensions, average fitness of 4 rabbits, and convergence curve.

Table 6
Significance comparisons of WSRT for ARO vs PSO, ASO, and DE.

Function	PSO vs ARO				ASO vs ARO				DE vs ARO			
	p-value	T+	T-	Winner	*p*-value	T+	T-	Winner	*p*-value	T+	T-	Winner
$F_1(x)$	1.73E−06	0	465	+	4.32E−08	0	465	+	1.73E−06	0	465	+
$F_2(x)$	1.73E−06	0	465	+	4.32E−08	0	465	+	1.73E−06	0	465	+
$F_3(x)$	1.73E−06	0	465	+	4.32E−08	0	465	+	1.73E−06	0	465	+
$F_4(x)$	1.73E−06	0	465	+	4.32E−08	0	465	+	1.73E−06	0	465	+
$F_5(x)$	1.73E−06	0	465	+	2.13E−06	2	463	+	0.047162	136	329	+
$F_6(x)$	0.25	0	465	=	1	0	465	=	0.25	0	465	=
$F_7(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_8(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_9(x)$	1.73E−06	0	465	+	4.32E−08	0	465	+	1.73E−06	0	465	+
$F_{10}(x)$	1.73E−06	0	465	+	4.32E−08	0	465	+	1.73E−06	0	465	+
$F_{11}(x)$	1.73E−06	0	465	+	1	0	465	=	1.73E−06	0	465	+
$F_{12}(x)$	1.73E−06	0	465	+	1.73E−06	465	0	–	0.000359	59	406	+
$F_{13}(x)$	0.000664	67	398	+	1.73E−06	465	0	–	1.80E−05	24	441	+
$F_{14}(x)$	1	0	465	=	1	0	465	=	1	0	465	=
$F_{15}(x)$	1.73E−06	0	465	+	1.01E−07	0	465	+	0.703247	59.5	405.5	=
$F_{16}(x)$	1	0	465	=	1	0	465	=	1	0	465	=
$F_{17}(x)$	1	0	465	=	1	0	465	=	1	0	465	=
$F_{18}(x)$	0.289063	9	456	=	4.32E−08	0	465	+	0.507813	30	435	=
$F_{19}(x)$	1	0	465	=	1	0	465	=	1	0	465	=
$F_{20}(x)$	0.000293	0	465	+	1	0	465	=	0.003906	0	465	+
$F_{21}(x)$	2.64E−05	0	465	+	4.32E−08	0	465	+	1	0	465	=
$F_{22}(x)$	0.000977	0	465	+	1	0	465	=	1	0	465	=
$F_{23}(x)$	6.10E−05	0	465	+	1	0	465	=	1	0	465	=
$F_{24}(x)$	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	+
$F_{27}(x)$	0.205888	294	171	=	1.92E−06	464	1	–	0.000664	398	67	–
$F_{28}(x)$	1.49E−05	22	443	+	0.010444	108	357	+	0.571646	205	260	=
$F_{29}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.92E−06	1	464	+
$F_{30}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	0.102011	153	312	=
$F_{31}(x)$	1.73E−06	0	465	+	1.64E−05	23	442	+	5.75E−06	453	12	–

Table 7
Statistical comparisons of WSRT for ARO vs CS, GSA, ABC, and TLBO.

Function	CS vs ARO				GSA vs ARO				ABC vs ARO				TLBO vs ARO			
	p-value	T+	T-	Winner	*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	+	1.73E−06	0	465	+
$F_2(x)$	1.73E−06	0	465	+	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	+	1.73E−06	0	465	+
$F_4(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	1.92E−06	1	464	+
$F_5(x)$	1.73E−06	0	465	+	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	=	1	0	465	=
$F_7(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	2.35E−06	3	462	+
$F_8(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	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	+	1.73E−06	0	465	+
$F_{10}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	8.12E−07	0	465	+
$F_{11}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	1	0	465	=
$F_{12}(x)$	1.73E−06	0	465	+	0.64351659	210	255	=	1.73E−06	0	465	+	3.11E−05	30	435	+
$F_{13}(x)$	0.001833	81	384	+	1.73E−06	465	0	–	1.73E−06	0	465	+	0.001593	79	383	+
$F_{14}(x)$	1	0	465	=	2.56E−06	0	465	+	1.73E−06	0	465	+	1	0	465	=
$F_{15}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	0.22658	75	390	=
$F_{16}(x)$	1	0	465	=	1	0	465	=	1	0	465	=	1	0	465	=
$F_{17}(x)$	1	0	465	=	1	0	465	=	1.73E−06	0	465	+	1	0	465	=
$F_{18}(x)$	0.387695	52	413	=	3.41E−07	0	465	+	1.73E−06	0	465	+	0.21875	3.5	462	=
$F_{19}(x)$	1	0	465	=	1	0	465	=	0.0078125	0	465	+	1	0	465	=
$F_{20}(x)$	2.10E−06	0	465	+	1	0	465	=	1	0	465	=	1	0	465	=
$F_{21}(x)$	0.0625	0	465	=	0.00012207	0	465	+	1.58E−06	0	465	+	1	0	465	=
$F_{22}(x)$	0.007813	0	465	+	1	0	465	=	3.27E−07	0	465	+	1	0	465	=
$F_{23}(x)$	8.16E−05	0	465	+	1	0	465	=	7.03E−07	0	465	+	1	0	465	=
$F_{24}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	1.72E−06	0	465	+
$F_{25}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{26}(x)$	1.73E−06	0	465	+	1.71E−06	0	465	+	1.73E−06	0	465	+	1	0	465	=
$F_{27}(x)$	9.71E−05	422	43	–	5.79E−05	37	428	+	0.0570965	325	140	=	0.81302	221	244	=
$F_{28}(x)$	0.000222	53	412	+	1.92E−06	1	464	+	4.73E−06	10	455	+	0.00014	47	418	+
$F_{29}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{30}(x)$	1.73E−06	0	465	+	0.51704789	264	201	=	1.73E−06	0	465	+	0.01108	109	356	+
$F_{31}(x)$	0.110926	155	310	=	2.35E−06	3	462	+	1.73E−06	0	465	+	0.00053	401	64	–

Table 8
Statistical results of WSRT.

Function type	PSO vs ARO (+/=/-)	ASO vs ARO (+/=/-)	DE vs ARO (+/=/-)	CS vs ARO (+/=/-)	GSA vs ARO (+/=/-)	ABC vs ARO (+/=/-)	TLBO vs ARO (+/=/-)
Unimodal	6/1/0	6/1/0	6/1/0	6/1/0	6/1/0	6/1/0	6/1/0
Multimodal	6/0/0	3/1/2	6/0/0	6/0/0	4/1/1	6/0/0	5/1/0
Low-dimension	5/5/0	3/7/0	1/9/0	4/6/0	4/6/0	8/2/0	0/10/0
Composition	7/1/0	7/0/1	4/2/2	6/1/1	7/1/0	7/1/0	5/2/1
Total	24/7/0	19/9/3	17/12/2	22/8/1	21/9/1	27/4/0	16/14/1

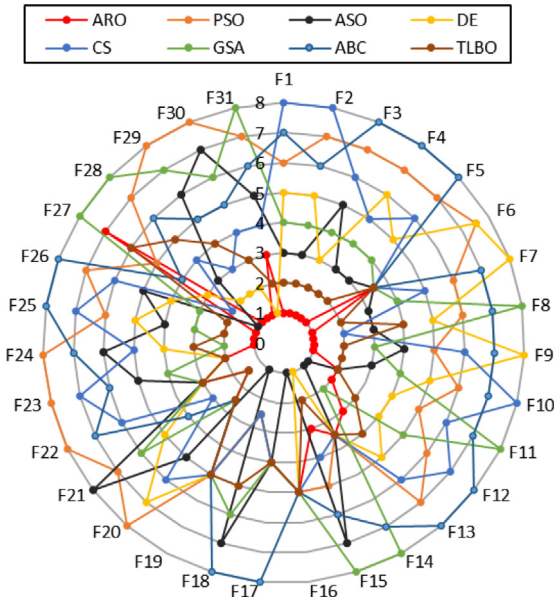


Fig. 18. Radar chart for ranks among all compared algorithms.

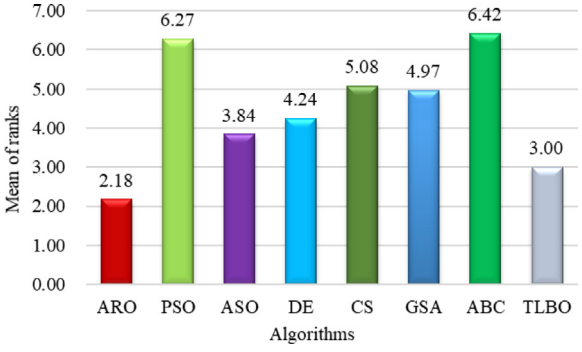


Fig. 19. Mean ranks obtained by Friedman test for 31 functions using various algorithms.

optimizers. It reveals there is still a significant gap in searching ability between ARO and other methods even when the dimensions of the questions are very high.

4.7. Comparison of ARO with some latest algorithms

To further test the performance of ARO, selected new optimizers proposed in recent years, including COA (Pierezan and Coelho, 2018), PBO (Połap and Woźniak, 2017), OOA (de Vasconcelos Segundo et al., 2019b), BWO (Hayyolalam and Kazem, 2020), and FOA (de Vasconcelos Segundo et al., 2019a), are compared with ARO. All initial parameters are set as the population size = 50, the maximum iterations = 1000. The results from all optimizers are based on 30 runs. Figs. 21 and 22 depict the bar charts from ARO and other optimizations on the benchmark functions described in Appendix. Based on the bar charts, ARO provides the best results on 12 of 31 functions (F1–F4, F9–F12, F24–F26, and F30) compared to the other 5 optimizers. The results of FOA are better than those of the other optimizers on 6 of 31 functions (F5–F7, F13, F28, and F29). COA outperforms the others on 3 of 31 functions (F8, F27, and F31). ARO achieves similar performance to that of one or more other optimizers on 10 of 31 functions (F14–F23). Most these optimizers have good optimization ability on the low-dimensional multimodal functions.

Tables 10 and 11 provide the significance comparisons of WSRT for ARO versus each of the other algorithms for each function, and the significance comparisons are listed in Table 12. It is clear that ARO has a higher number of '+' than the other optimizers, which reveals the statistically significance difference between ARO and each of the other competitors. It means that ARO shows better performance for the functions with '+' . Meanwhile, it can be found that there is a high number of '=' when ARO is compared with COA and FOA, respectively. It shows that the difference is not statistically significant between ARO and the two optimizers and ARO has similar performance to COA and FOA for the functions with '=', respectively. Fig. 23 illustrates the radar chart for the ranks by the Friedman test among these algorithms. It can be observed that ARO provides a better rank than the other optimizers for most functions. Fig. 24 depicts the mean ranks obtained by the Friedman test for 31 functions using ARO and the 5 algorithms. We can observe that ARO ranks first, followed by FOA, COA, OOA, BWO, and PBO. The results show that, although FOA and COA perform well, ARO shows significant competition among the latest optimizers.

4.8. Application of ARO in semi-real engineering problems

In this subsection, five semi-real engineering problems are employed. ARO needs to incorporate a constraint-handing technique to solve these problems. There are various constraint-handing techniques such as penalty functions, repair algorithms, decoder functions, feasibility preserving representations, and operators (Kramer, 2010). Among them, penalty functions are the most popular constraint-handling methods because of their simplicity and ease of implement. For simplicity, a common penalty method is used to tackle constraints for these semi-real engineering problems (Coello, 2002).

$$\text{Minimize } F(\vec{x}) = f(\vec{x}) \pm \left(\sum_{i=1}^{p} a_i G_i(\vec{x}) + \sum_{j=1}^{q} b_j H_j(\vec{x})\right) \tag{24}$$

$$G_i(\vec{x}) = \max(0, g_i(\vec{x}))^{\eta} \tag{25}$$

$$H_j(\vec{x}) = |h_j(\vec{x})|^{\lambda} \tag{26}$$

where $g_i(\vec{x})$ is the inequality constraint, $h_i(\vec{x})$ is the equality constraint, p is the number of inequality constraints and q is the number of equality constraints, a_i and b_j are positive constants, respectively, and η and λ are 1 or 2. For this penalty method, the objective function value will increase when a candidate solution violates any constraint, resulting in that algorithms are pushed inside the feasible region from the infeasible one.

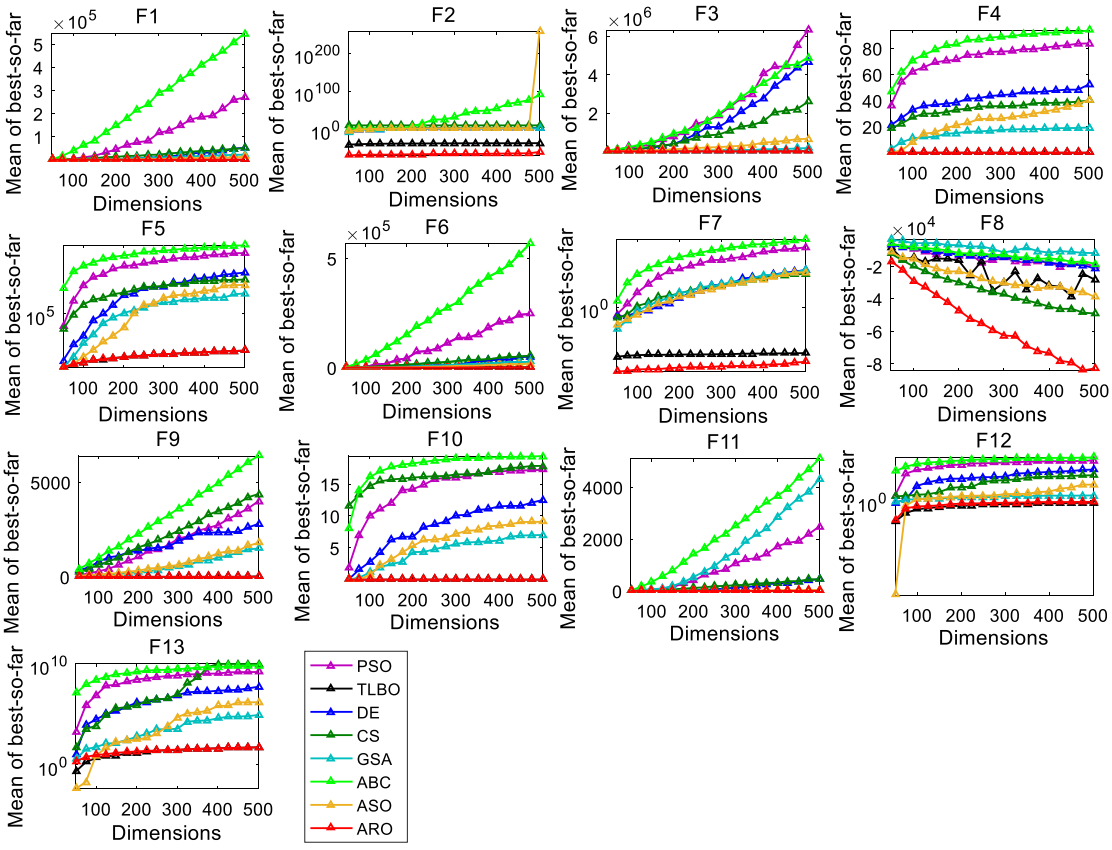


Fig. 20. Scalability results of ARO versus other optimizers on the scalable functions.

Table 9
Results of all optimizers on functions F1–F13 with 500 dimensions.

Function	Index	ARO	PSO	ASO	DE	CS	GSA	ABC	TLBO
$F_1(x)$	Mean	9.21E−111	2.61E+05	1.25E+04	5.44E+04	4.69E+04	2.31E+04	5.43E+05	4.94E−74
	Std	2.80E−111	1.73E+04	1.67E+03	2.17E+03	4.74E+03	2.50E+03	1.44E+04	3.41E−74
$F_2(x)$	Mean	1.45E−61	7.12E+02	6.52E+262	2.93E+02	1.00E+10	6.34E+250	3.89E+93	1.22E−37
	Std	4.11E−61	2.84E+01	9.26E+263	2.23E+01	3.42E+09	1.92E+250	8.69E+93	4.34E−38
$F_3(x)$	Mean	1.89E−82	4.55E+06	6.27E+05	3.97E+06	2.93E+06	1.54E+05	5.44E+06	1.39
	Std	1.08E−81	1.43E+06	5.59E+04	7.59E+05	2.25E+05	4.27E+04	5.42E+05	1.02
$F_4(x)$	Mean	3.35E−44	8.16E+01	3.97E+01	5.09E+01	3.75E+01	1.90E+01	9.27E+01	2.44E−29
	Std	7.35E−44	3.85	4.22	5.03	2.07	6.17E−01	0.98	3.93E−29
$F_5(x)$	Mean	3.38	5.38E+08	5.66E+06	3.00E+07	1.40E+07	1.75E+06	1.75E+09	4.96E+02
	Std	3.12	7.57E+07	1.10E+06	4.32E+06	2.68E+06	3.70E+05	1.18E+08	4.65E−01
$F_6(x)$	Mean	0	2.66E+05	1.92E+04	5.37E+04	5.52E+04	2.57E+04	5.62E+05	0
	Std	0	2.57E+04	2.87E+03	5.22E+03	2.67E+03	2.59E+03	2.33E+04	0
$F_7(x)$	Mean	3.12E−04	3.86E+03	1.15E+02	2.03E+02	8.81E+01	1.64E+02	1.25E+04	2.07E−03
	Std	1.08E−04	4.20E+02	1.54E+01	3.71E+01	8.975318968	3.48E+01	1.12E+03	7.34E−04
$F_8(x)$	Mean	−8.58E+04	−2.03E+04	−3.46E+04	−2.00E+04	−5.00E+04	−1.17E+04	−1.83E+04	−3.07E+04
	Std	1.57E+03	1.70E+03	4.10E+03	1.47E+03	1.66E+03	1.36E+03	1.79E+03	1.50E+04
$F_9(x)$	Mean	0	3.81E+03	1.77E+03	2.61E+03	4.43E+03	1.54E+03	6.32E+03	0
	Std	0	1.41E+02	1.11E+02	7.50E+02	7.00E+01	8.60E+01	8.48E+01	0
$F_{10}(x)$	Mean	8.88E−16	1.74E+01	9.46E+00	1.24E+01	1.67E+01	6.90E+00	1.93E+01	9.41E−15
	Std	0	3.86E−02	4.07E−01	4.47E−01	8.32E−01	2.79E−01	1.58E−01	3.18E−15
$F_{11}(x)$	Mean	0	2.26E+03	4.74E+01	4.69E+02	4.54E+02	4.40E+03	5.00E+03	2.22E−17
	Std	0	1.00E+02	1.38E+01	2.96E+01	3.13E+01	1.22E+02	2.34E+02	4.97E−17
$F_{12}(x)$	Mean	5.45E−03	7.67E+08	3.48E+03	4.62E+06	6.29E+05	1.28E+01	3.45E+09	3.19E−01
	Std	1.82E−03	1.05E+08	6.62E+03	5.69E+05	3.32E+05	2.35E+00	2.58E+08	3.63E−02
$F_{13}(x)$	Mean	1.46	1.65E+09	1.83E+04	5.13E+07	1.00E+10	8.55E+04	6.85E+09	4.97E+01
	Std	0.98	3.90E+08	1.85E+04	1.68E+07	2.13E+09	6.29E+04	7.19E+08	1.16E−02

4.8.1. Pressure vessel design

The objective of this problem is to minimize the fabrication cost of a pressure vessel shown in Fig. 25 (Kannan and Kramer, 1994). There are four variables to be optimized. There are four constraints that need to be met in this problem. The problem formulation is presented below.

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

Minimize $f_1(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$.

Subject to

$g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0,$

$g_2(\vec{x}) = -x_2 + 0.00954x_3 \le 0,$

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

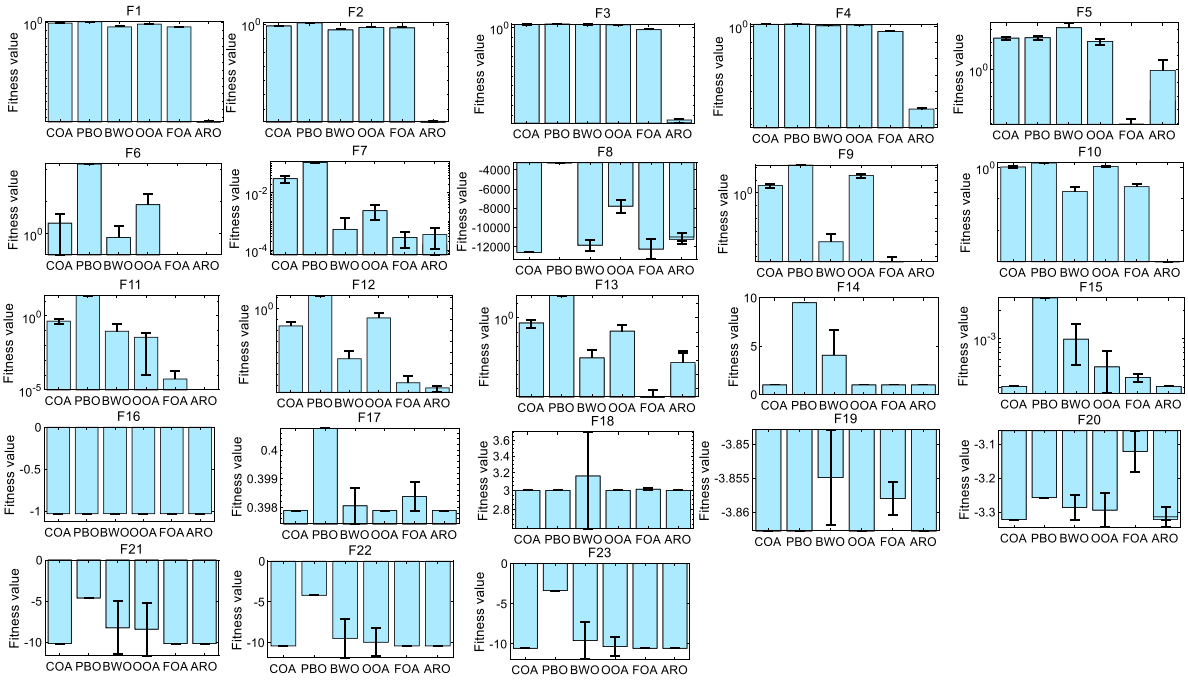


Fig. 21. Bar charts provided by ARO and latest algorithms on 23 functions.

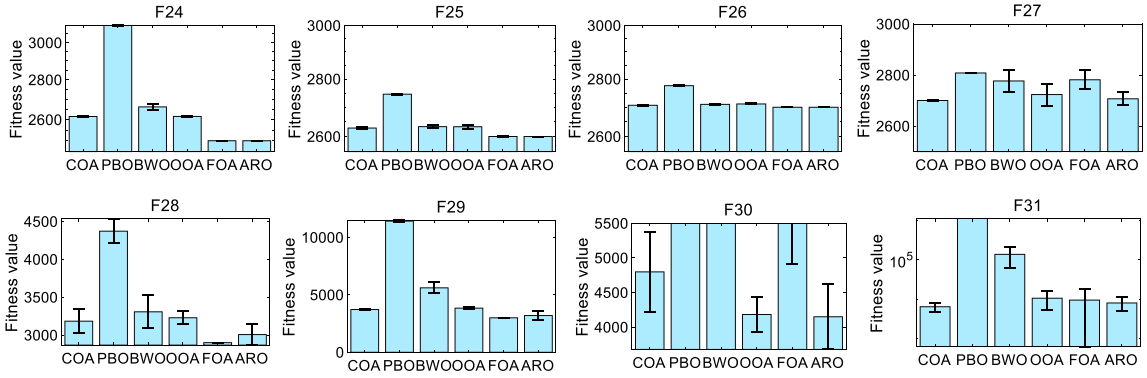


Fig. 22. Bar charts provided by ARO and latest algorithms on 8 composition functions.

$g_4(\vec{x}) = x_4 - 240 \le 0,$

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

This problem is optimized by ARO and the results are compared with the results provided by MFO (Mirjalili, 2015), GA (Coello and Montes, 2002), CS (Gandomi et al., 2013), CPSO (He and Wang, 2007a), HPSO (He and Wang, 2007b), (μ+λ)ES (Mezura-Montes and Coello, 2005), CDE (Huang et al., 2007), ACO (Kaveh and Talatahari, 2010b), ABC (Akay and Karaboga, 2012), and DE (Li et al., 2007) in the literature. The statistical results of the optimal values and the optimal variables obtained by these optimizers are presented in Table 13. ARO can obtain the optimal function value $f_1 = 5885.667948$ with the structure variables $x = (0.77824311, 0.38475065, 40.32338898, 199.94794222)$. The convergence curve and constraint values provided by ARO are revealed in Fig. 26. The results suggest that ARO finds a very different structure compared to others which can achieve the minimum fabrication cost.

4.8.2. Rolling element bearing design

This engineering case in Fig. 27 is the maximum of the dynamic load-carrying capacity of the rolling element bearing (Rao and Tiwari, 2007). There are ten constraints and ten design variables. The problem formulation is presented below.

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

$$\text{Maximize} \begin{cases} f_2(\vec{x}) = f_c Z^{2/3} D_b^{1.8} & if\ D_b \le 25.4\ \text{mm} \\ f_2(\vec{x}) = 3.647 f_c Z^{2/3} D_b^{1.4} & if\ D_b > 25.4\ \text{mm} \end{cases},$$

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

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

Table 10
Significance comparisons of WSRT for ARO vs COA, PBO, and BWO.

Function	COA vs ARO				PBO vs ARO				BWO vs ARO			
	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	+
$F_6(x)$	2.37E−05	0	465	+	1.73E−06	0	465	+	2.44E−04	0	465	+
$F_7(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	3.29E−01	185	280	=
$F_8(x)$	1.73E−06	465	0	−	1.73E−06	0	465	+	1.24E−05	445	20	−
$F_9(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	1.71E−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	+	3.82E−01	275	190	=
$F_{13}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+	8.29E−01	222	243	=
$F_{14}(x)$	1	0	465	=	1.73E−06	0	465	+	3.79E−06	0	465	+
$F_{15}(x)$	5.41E−02	139	326	=	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{16}(x)$	1	1	464	=	1	0	465	=	1	49	416	=
$F_{17}(x)$	1	0	465	=	1.73E−06	0	465	+	1	0	465	=
$F_{18}(x)$	1.58E−01	75.5	389.5	=	1	0	465	=	1.95E−03	0	465	+
$F_{19}(x)$	1	0	465	=	1	0	465	=	1.73E−06	0	465	+
$F_{20}(x)$	2.50E−01	6	459	=	1.73E−06	0	465	+	2.61E−04	55	410	+
$F_{21}(x)$	4.53E−01	20	445	=	1.73E−06	0	465	+	2.06E−05	1	464	+
$F_{22}(x)$	1	49	416	=	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{23}(x)$	1.01E−07	0	465	+	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{24}(x)$	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	0	465	=
$F_{27}(x)$	7.04E−01	214	251	=	1.73E−06	0	465	+	1.49E−05	22	443	+
$F_{28}(x)$	2.41E−04	54	411	+	1.73E−06	0	465	+	1.80E−05	24	441	+
$F_{29}(x)$	2.37E−05	27	438	+	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{30}(x)$	8.19E−05	41	424	+	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{31}(x)$	1.48E−02	351	114	−	1.73E−06	0	465	+	1.73E−06	0	465	+

Table 11
Significance comparisons of WSRT for ARO vs OOA and FOA.

Function	OOA vs ARO				FOA vs ARO			
	p-value	T+	T-	Winner	*p*-value	T+	T-	Winner
$F_1(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_2(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_3(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_4(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_5(x)$	1.73E−06	0	465	+	1.73E−06	465	0	−
$F_6(x)$	2.48E−06	0	465	+	1	0	465	=
$F_7(x)$	1.73E−06	0	465	+	1.71E−01	299	166	=
$F_8(x)$	1.73E−06	0	465	+	3.59E−04	59	406	+
$F_9(x)$	1.73E−06	0	465	+	1.73E−06	465	0	−
$F_{10}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{11}(x)$	1.73E−06	0	465	+	2.84E−03	255	210	−
$F_{12}(x)$	1.73E−06	0	465	+	7.04E−01	251	214	=
$F_{13}(x)$	1.73E−06	0	465	+	8.13E−01	221	244	=
$F_{14}(x)$	1	0	465	=	1.73E−06	465	0	=
$F_{15}(x)$	4.29E−06	9	456	+	1.73E−06	0	465	+
$F_{16}(x)$	1	1	464	=	1.73E−06	0	465	+
$F_{17}(x)$	1	0	465	=	1.73E−06	0	465	+
$F_{18}(x)$	1.56E−01	2.5	462.5	=	1.73E−06	465	0	=
$F_{19}(x)$	1	0	465	=	1.73E−06	465	0	=
$F_{20}(x)$	8.26E−01	72	393	=	1.73E−06	0	465	+
$F_{21}(x)$	4.10E−02	10	455	+	1.73E−06	465	0	=
$F_{22}(x)$	2.84E−03	210	255	+	1.73E−06	465	0	=
$F_{23}(x)$	4.16E−06	29	436	+	1.73E−06	465	0	=
$F_{24}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{25}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{26}(x)$	1.73E−06	0	465	+	1.73E−06	0	465	+
$F_{27}(x)$	8.61E−01	241	224	=	3.52E−06	7	458	+
$F_{28}(x)$	1.49E−05	22	443	+	1.24E−05	445	20	−
$F_{29}(x)$	5.75E−06	12	453	+	2.84E−03	255	210	−
$F_{30}(x)$	7.34E−01	216	249	=	1.73E−06	0	465	+
$F_{31}(x)$	2.70E−02	125	340	+	7.19E−01	215	250	=

Table 12
Statistical results of WSRT.

Function type	COA vs ARO (+/=/-)	PBO vs ARO (+/=/-)	BWO vs ARO (+/=/-)	OOA vs ARO (+/=/-)	FOA vs ARO (+/=/-)
unimodal	7/0/0	7/0/0	6/1/0	7/0/0	4/2/1
multimodal	5/0/1	6/0/0	3/2/1	6/0/0	2/2/2
Low-dimension	1/9/0	7/3/0	8/2/0	4/6/0	4/6/0
Composition	6/1/1	8/0/0	7/1/0	6/2/0	5/1/2
Total	19/10/2	28/3/0	24/6/1	23/8/0	15/11/5

Table 13
Comparison results provided by various optimizers for pressure vessel design.

Algorithms	Optimal variables				Optimal cost
	T_s	T_h	R	L	
ARO	0.77824311	0.38475065	40.32338898	199.94794222	5885.667948
MFO	0.8125	0.4375	42.0984	176.6365	6059.7143
CS	0.8125	0.4375	42.0984	176.6366	6059.7143
GA	0.8125	0.4375	42.097398	176.6540	6059.9463
CPSO	0.8125	0.4375	42.091266	176.7465	6061.0777
HPSO	0.8125	0.4375	42.098400	176.6366	6059.7143
CDE	0.8125	0.4375	42.098411	176.6377	6059.7340
ABC	0.8125	0.4375	42.098446	176.6366	6059.7143
$(\mu+\lambda)$ES	0.8125	0.4375	42.098446	176.6366	6059.7143
DE	0.8125	0.4375	42.098411	176.637690	6059.7340
ACO	0.8125	0.4375	42.103624	176.572656	6059.0888

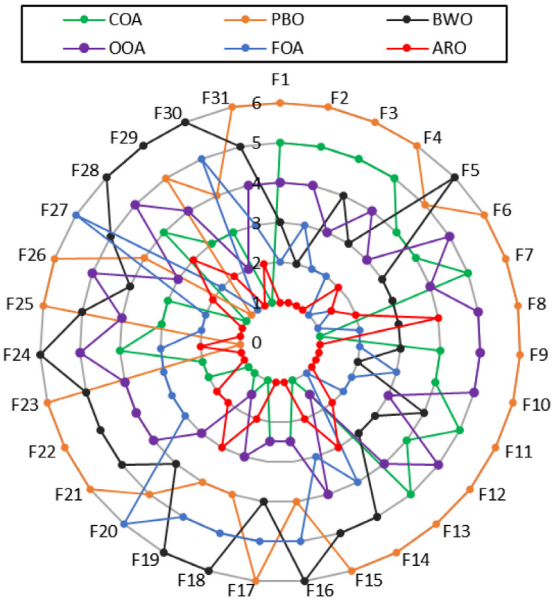


Fig. 23. Radar chart for ranks from ARO and latest algorithms.

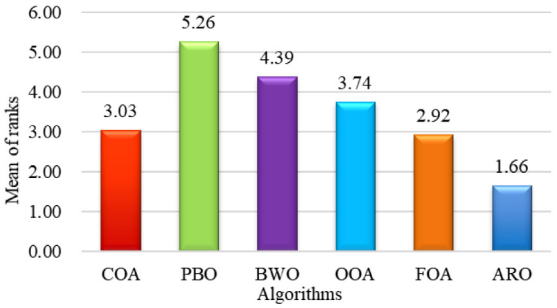


Fig. 24. Mean ranks obtained by Friedman test for 31 functions from ARO and latest algorithms.

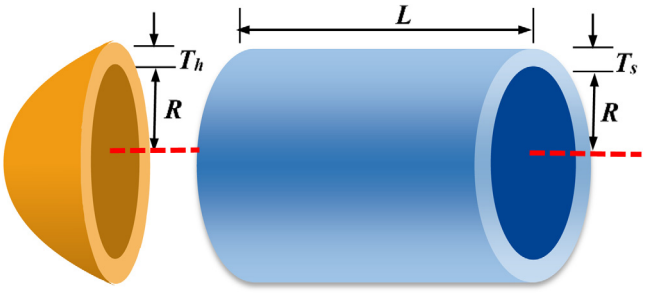


Fig. 25. Pressure vessel design.

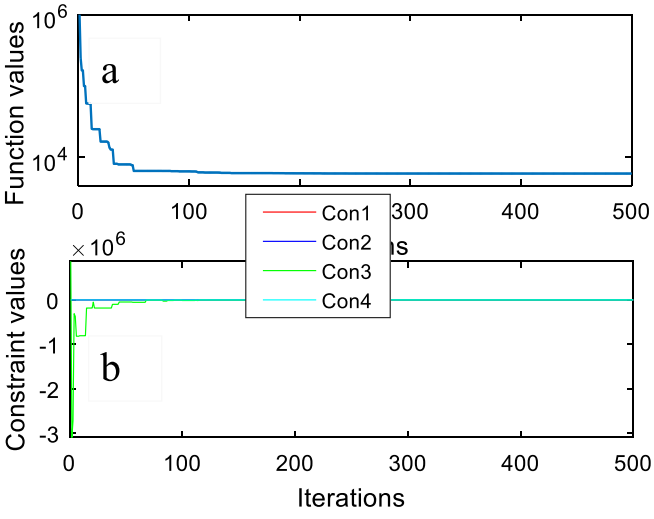


Fig. 26. (a) Convergence curve, (b) constraint values versus iterations.

Table 14
Comparison results provided by various optimizers for rolling element bearing design.

Algorithms	Optimal variables										Optimal load-carrying capacity
	D_m	D_b	Z	f_i	f_o	K_{Dmin}	K_{Dmax}	ε	e	ζ	
ARO	125.7189	21	10.5403	0.5150	0.5150	0.4459	0.672132	0.3000	0.0825	0.6317	85548.5106
GA2	125.7171	21.4231	11	0.5150	0.5150	0.4159	0.6510	0.3000	0.0223	0.7510	81843.3000
TLBO	25.7191	21.4256	11	0.5150	0.5150	0.4242	0.6339	0.3000	0.0689	0.7995	81859.7400
MBA	125.7153	21.4233	11	0.5150	0.5150	0.4888	0.6278	0.3002	0.0946	0.6461	85535.9611
PVS	125.7191	21.4256	11	0.5150	0.5150	0.4004	0.6802	0.3000	0.0800	0.7000	81859.74121
SOA	125	21.4189	11	0.5150	0.5150	0.4	0.7	0.3	0.02	0.6	85068.0520
PSO	125.7251	21	11.2924	0.5153	0.5816	0.4569	0.6540	0.3092	0.0256	0.6128	81534.2759
DE	125.7192	21.2508	10.8654	0.5153	0.56021	0.4199	0.6197	0.3012	0.0472	0.6740	83629.26366

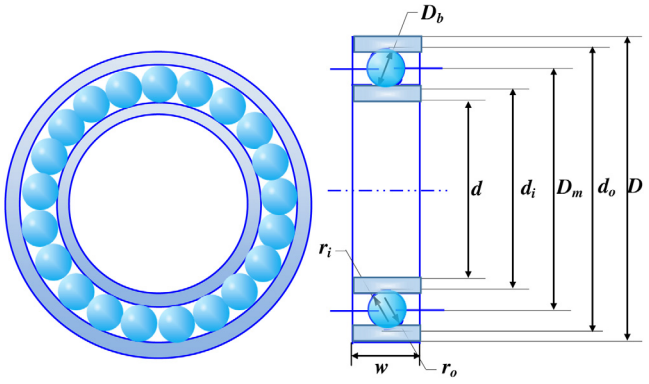


Fig. 27. Pressure vessel design.

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

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

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

Variable range

$$0.5(D+d) \le D_m \le 0.6(D+d), 0.15(D-d) \le D_b \le 0.45(D-d), 4 \le Z \le 50,$$

$$0.515 \le f_i \le 0.6, 0.515 \le f_o \le 0.6, 0.4 \le K_{D\min} \le 0.5, 0.6 \le K_{D\max} \le 0.7, 0.3 \le \varepsilon \le 0.4,$$

$$0.02 \le e \le 0.1, 0.6 \le \zeta \le 0.85.$$

This problem is tackled by many optimizers such as PVS (Savsani and Savsani, 2016), TLBO (Rao et al., 2011), GA2 (Savsani and Savsani, 2016), SOA (Dhiman and Kumar, 2019), and MBA (Sadollah et al., 2013), and the results are compared with those from our optimizer. Table 14 gives the comparisons of the optimal variables and function values. It can be found that ARO provides better results than its competitors when solving this problem. Fig. 28 gives the convergence curve and constraint values of ARO.

4.8.3. Tension/compression spring design

This problem aims to minimize the weight of the coil and meet the requirement of three constraints. The pictorial representation of this problem is given in Fig. 29, in which three variables need to be optimized below.

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

Minimize $f_3(\vec{x}) = (x_3 + 2)x_2x_1^2$.

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

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

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0, g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0.$$

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

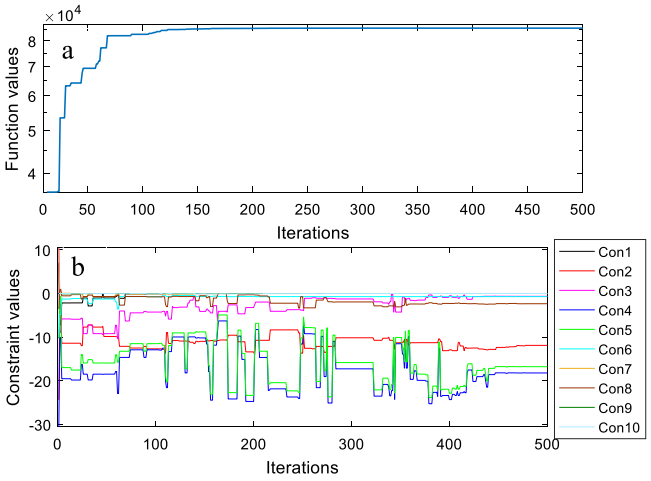


Fig. 28. (a) Convergence curve, (b) constraint values versus iterations.

The comparisons between ARO and some competitors such as CPSO (He and Wang, 2007a), HPSO (He and Wang, 2007b), MFO (Mirjalili, 2015), HS (Mahdavi et al., 2007), CDE (Huang et al., 2007), ES (Mezura-Montes and Coello, 2008), (μ+λ)ES (Mezura-Montes and Coello, 2005) are presented in Table 15. ARO can obtain the optimal function value f_3 = 0.01266602 with the structure variables x = (0.05189732, 0.36174867, 11). Although the optimal structure of coil belongs to HPSO, ARO provides satisfactory results compared to other algorithms. Fig. 30 gives the convergence curve and constraint values which unveils that ARO can find the global optimum with high efficiency when solving this problem.

4.8.4. Cantilever beam design

This problem deals in Fig. 31 with minimizing the weight of a cantilever beam subject to one constraint, and there are five variables which represent five different block lengths. The problem formulation is presented below.

Consider variable $\vec{x} = [x_1, x_2, x_3, x_4, x_5]$.

Minimize $f_4(\vec{x}) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5)$.

Subject to $g_1(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0.$

Variable range $0.01 \le x_i \le 100,\ i = 1, \ldots, 5.$

The optimal results from ARO and other competitors such as GOA (Saremi et al., 2017), SOS (Cheng and Prayogo, 2014), GCAI (Nowcki, 1974), MFO (Mirjalili, 2015), CS (Gandomi et al., 2013), GCAII (Nowcki, 1974) and MMA (Chickermane and Gea, 1996) are tabulated in Table 16. ARO can provide the optimal function value f_4 = 1.33996 with the structure variables x = (6.00682926, 5.31143662, 4.49352431, 3.50289770, 2.15904513). From Table 16, ARO obtains the same best objective value as GOA and SOS, but their obtained

Table 15
Comparison results provided by various optimizers for tension/compression spring design.

Algorithms	Optimal variables			Optimal weigh
	d	D	N	
ARO	0.05189732	0.36174867	11	0.01266602
CPSO	0.051728	0.357644	11.244543	0.0126747
HPSO	0.051706	0.357126	11.265083	0.0126652
MFO	0.051994457	0.36410932	10.868421862	0.0126669
HS	0.051154	0.349871	12.076432	0.0126706
CDE	0.051609	0.354714	11.410831	0.0126702
ES	0.051989	0.363965	10.890522	0.0126810
$(\mu+\lambda)$ES	0.052836	0.384942	9.807729	0.012689

Table 16
Comparison results provided by various optimizers for cantilever beam design.

Algorithms	Optimal variables					Optimal weigh
	x_1	x_2	x_3	x_4	x_5	
ARO	6.00682926	5.31143662	4.49352431	3.50289770	2.15904513	1.33996
SOS	6.01878	5.30344	4.49587	3.49896	2.15564	1.33996
CS	6.00890	5.30490	4.50230	3.50770	2.15040	1.33999
GCAI	6.01000	5.30000	4.49000	3.49000	2.15000	1.34000
GCAII	6.01000	5.30000	4.49000	3.49000	2.15000	1.34000
MMA	6.01000	5.30000	4.49000	3.49000	2.15000	1.34000
MFO	5.984871	5.316726	4.497332	3.513616	2.16162	1.339988
GOA	6.011674	5.31297	4.48307	3.50279	2.16333	1.33996

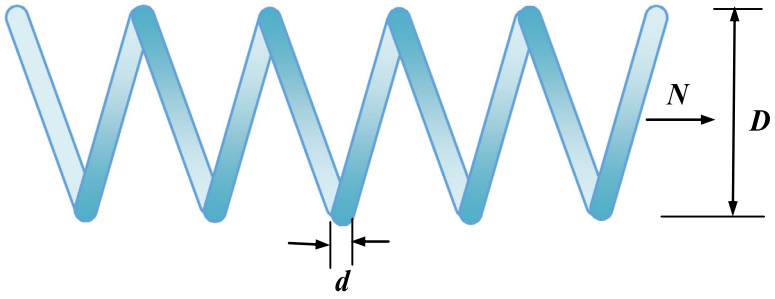


Fig. 29. Tension/compression string design.

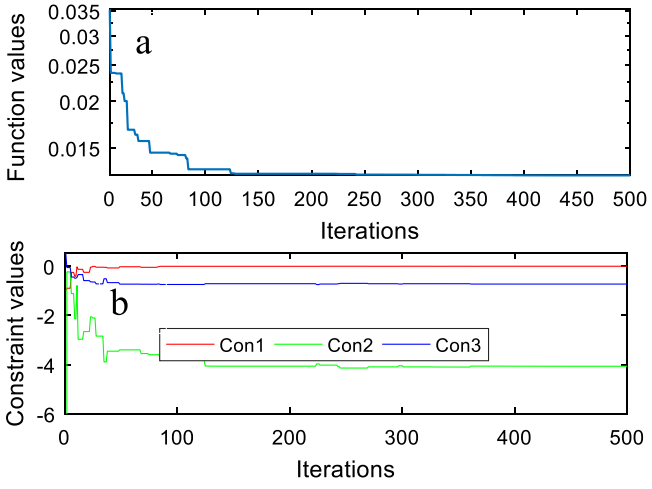


Fig. 30. (a) Convergence curve, (b) constraint values versus iterations.

optimal variables are different. Therefore, ARO can find other optimal structure parameters for this design. The convergence curve and constraint values of ARO for this design are given in Fig. 32.

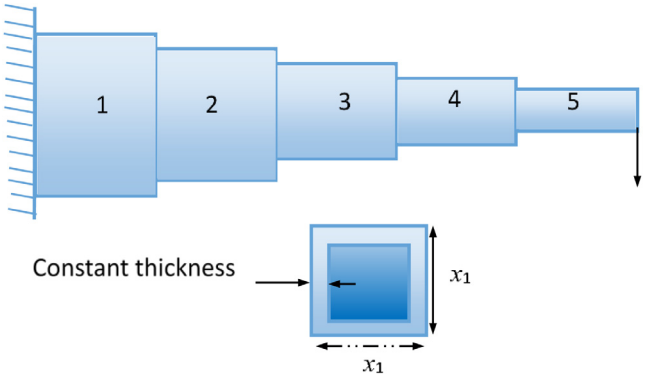


Fig. 31. Cantilever beam design.

4.8.5. Gear train design

The problem aims to minimize the cost of gear ratio of the gear train shown in Fig. 33 (Rao et al., 2011). There are four integer variables in this problem, in which T_a, T_b, T_d, and T_f represent the teeth number of four different gearwheels. The gear ratio is given as $T_b/T_a \cdot T_d/T_f$. The problem formulation is presented below.

variable Consider $\vec{x} = [x_1, x_2, x_3, x_4] = [T_a, T_b, T_d, T_f]$

Minimize $f_5(\vec{x}) = (\frac{1}{6.931} - \frac{T_b \cdot T_d}{T_a \cdot T_f})^2$.

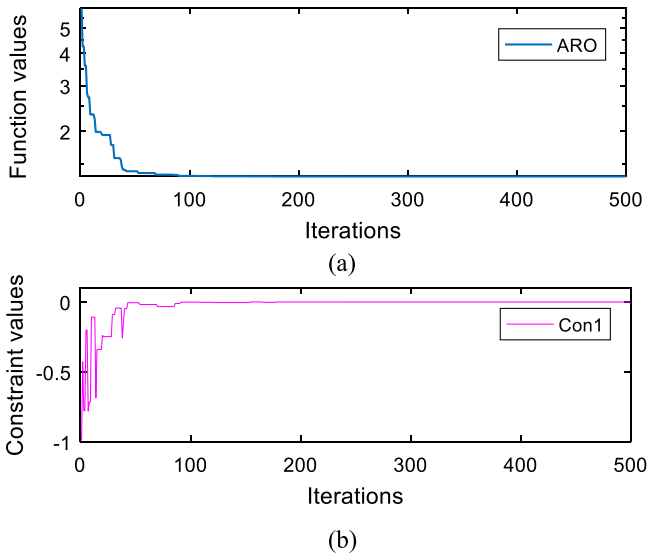


Fig. 32. (a) Convergence curve, (b) constraint values versus iterations.

Subject to (variable range) $0.01 \leq x_1, x_2, x_3, x_4 \leq 60$.

The comparison results obtained by ARO and other optimizers, such as MFO (Mirjalili, 2015), WOA (Mirjalili and Lewis, 2016), ABC (Sadollah et al., 2013), MBA (Sadollah et al., 2013), GA3 (Deb and Goyal, 1996), CS (Gandomi et al., 2013), PSO, and DE, are given in Table 17. It can be observed that the optimal design parameters obtained by ARO is x = (49, 19, 16, 43) with the objective value $f_5 = 2.7009\text{E-}012$. According to Table 14, the results suggest that ARO is very effective in searching for the optimal solution for this problem as well. ARO finds the same minimal cost as MFO, ABC, MBA, GA3, and CS. However, the optimal teeth numbers of gearwheels provided are different. The results reveal that ARO and other optimizers can provide another optimal alternative for this design. The convergence curve and variable values versus iterations from ARO and other optimizers for this design problem are given in Fig. 34.

5. Application of ARO in fault diagnosis of rolling bearing

The artificial neural network (ANN) is a popular machine learning technique. Due to its advantages such as the ability to learn by themselves, memory function, nonlinear mapping, and classification and recognition, ANN has been widely applied in different fields, including deep learning, image processing, control engineering, signal processing, expert system, and fault diagnosis. Recently, with the development of optimization technologies, combining optimization technologies with ANN makes it more powerful in tackling real applications. The BP network, a well-known multilayer feedforward neural network trained according to an error, is a frequently used multi-classification method in various engineering fields (Bebis and Georgiopoulos, 1994; Ma et al., 2020; Yu and Xu, 2014; Wu et al., 2019). In this part, ARO is merged into the BP network for the fault diagnosis of a rolling bearing. The decision variables are weights and threshold of the BP network, the objective function is the training errors between the desired output and the model output. Therefore, the fault diagnosis of a rolling bearing is defined as a minimization problem by which the optimal weights and threshold of the BP network can be obtained by minimizing the training errors. During the optimizing process, at each iteration, the optimal solutions obtained so far from ARO are passed to the BP network as the weights and threshold, and the training data are input into the BP network for training, then the output is obtained for calculating the training error via the desired output. When the optimizing process is completed, the optimal weights and threshold are saved into the BP network.

Table 17
Comparison results provided by various optimizers for gear train design.

Algorithms	Optimal variables				Optimal cost
	T_a	T_b	T_d	T_f	
ARO	49	19	16	43	2.7009E−12
MFO	43	19	16	49	2.7009E−12
ABC	49	16	19	43	2.7009E−12
MBA	43	16	19	49	2.7009E−12
GA3	49	16	19	43	2.7009E−12
CS	43	16	19	49	2.7009E−12
WOA	47	12	13	23	9.92157E−10
PSO	52	15	29	58	2.35764E−9
DE	47	19	15	42	9.5209E−9

The bearing fault experiment is conducted by using our optimizer and the experimental devices are depicted in Fig. 35. The experimental equipment consists of a motor, a coupling, and a roller bearing, and the rolling bearing used is of Type 6205. As shown in Fig. 35, two channels are utilized to collect data from the acceleration sensors both horizontally and vertically. The bearing defects are created by using the spark machining, with the depths of 1.0, 0.6, and 0.2 mm, respectively. the sampling frequency is taken as 10240 Hz. The rotation speeds with 1750, 1180, and 1478 rpms are monitored by the torque sensor. The bearing faults include the inner race, outer race, and rolling ball faults, and three vibration signals are from the bearings in each fault. Each vibration signal has 102,400 data points, the first 10.240 data points are deleted and the rest are divided into 90 samples for training and 30 samples for testing.

Fig. 36 shows the model of the bearing fault diagnosis based on the BP network optimized by ARO, which is referred to simply as ARO-BP. In this model, the diagnosis feature vectors are firstly extracted and constructed by the Empirical Mode Decomposition (EMD). The EMD can self-adaptively convert nonlinear and non-stationary signals into mono symmetric components (Junsheng et al., 2006). It may break down a signal into its components called Intrinsic Mode Functions (IMFs). Thus, these IMFs contain main fault information when the bearing fault signals are decomposed using EMD. In this experiment, the original bearing vibration signals are decomposed into nine IMFs, so the energy of different IMFs of every data sample as the feature vector of fault diagnosis model $F = [F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9]$ can be calculated as:

$$F_i = \sum_{k=1}^{1024} |c_i(k)|^2 i = 1, \ldots, 9 \tag{27}$$

where $c_i(k)$ is the ith IMF component at the time k. Fig. 37 shows the extraction process of feature vectors from an original outer race fault signal by the EMD decomposition. Obviously, the EMD method can convert a high-dimension data sample into a lower-dimensional feature vector, which can reduce the number of the input layer neurons in the ARO-BP model and improve its diagnosis efficiency.

The BP network adjusts the number of the hidden layer neurons and the weights between the two layers, and the threshold by minimizing the difference between the actual input and expected input (Bohat and Arya, 2018). Fig. 38 shows the architecture of the BP network. The hidden neuron is given:

$$H_j = \frac{1}{1 + e^{-q_j}} \tag{28}$$

$$q_j = \sum_{i=1}^{m} w_{ij} x_i - b_j j = 1, \ldots, h \tag{29}$$

where x_i is the ith variable of the input layer, m is the number of input layer neurons, h is the number of the hidden layer neurons, w_{ij} is the weight connecting the ith input layer neuron and the jth hidden layer neuron, and b_j is the threshold of the jth hidden layer neuron.

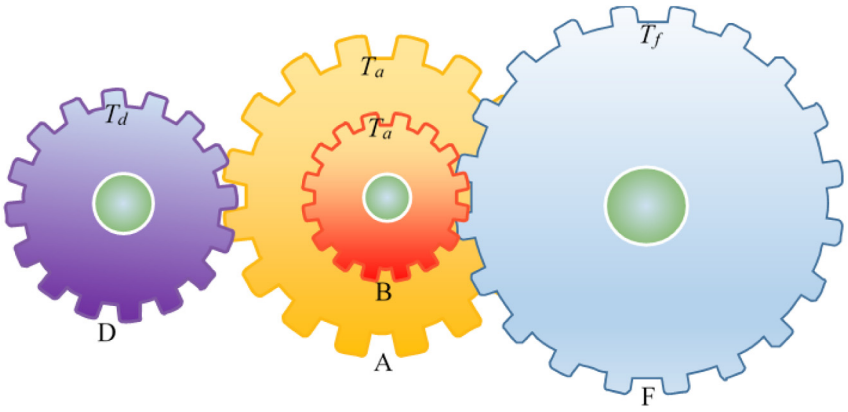


Fig. 33. Gear train design.

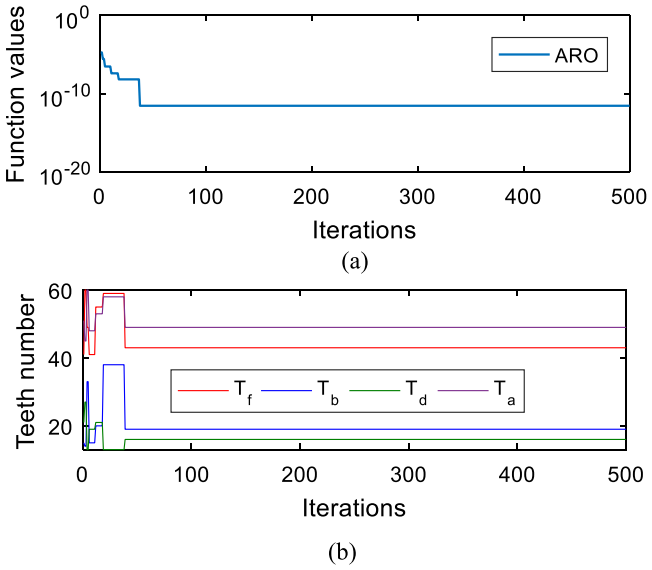


Fig. 34. (a) Convergence curve, (b) Variable values versus iterations.

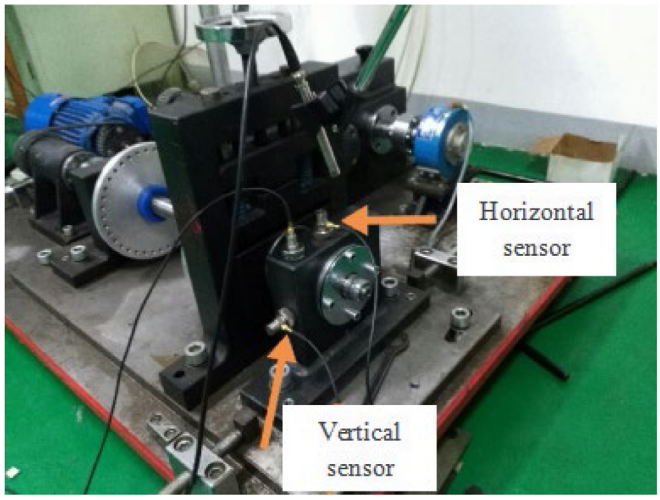


Fig. 35. Bearing fault diagnosis test equipment.

The output layer neuron is given by:

$$O_k = \frac{1}{1+e^{-o_k}} \tag{30}$$

$$o_k = \sum_{j=1}^{h} w'_{jk} H_j - b'_k k = 1, \ldots, s \tag{31}$$

where w_{jk} is the weight connecting the jth hidden layer neuron and the kth output layer neuron, b'_k is the threshold of the kth output neuron, and s is the number of the output layer neurons.

To achieve good performance for both training and testing data, a fitness function named Average Mean Square Error (AMSE) is established by tuning the values of weighs and threshold to minimize the difference between the output produced by BP network and the desired output:

$$\text{Minimize } f_{AMSE}(X) = \sum_{i=1}^{N}\sum_{j=1}^{s} \frac{(O_j^i - \hat{O}_j^i)^2}{N} \tag{32}$$

$$X = [w_{11}, \ldots, w_{mh}, b_1, \ldots, b_h, w'_{11}, \ldots, w'_{hs}, b'_1, \ldots, b'_s] \tag{33}$$

where N is the number of training samples, O_j^i and $\hat{O}_j^i$ are the desired output and model output, respectively, X is the variables containing the values of weights and threshold of the BP network, the dimension of X is (mh+h+hs+s), so X becomes the formation of each search individual in ARO-BP as shown in Fig. 38.

To verify the proposed model in tackling the bearing fault diagnosis problem, a comprehensive experiment with the defeat depth =0.6 mm is firstly conducted. The feature vectors of the training and testing sets are extracted by the EMD method and inputted to the BP network training, in which the weighs and threshold are optimized by the above-mentioned optimizers for comparisons. Meanwhile, the BP network with gradient descent (GD-BP) is also applied to this study. The feature vectors of three types of bearing faults used for the BP network are tabulated in Table 18. The results provided by these methods are compared with that of ARO. As there is no rule to determine the number of the hidden layer neurons, we consider the rule proposed by Heaton (2008), in which the number of the hidden neurons should be 2/3 the number of input neurons, plus the number of output neurons. Based on this rule, the number of hidden neurons is set to 9, thus the dimension of the solution space in the optimizers is 120.

Fig. 39 shows the convergence curves of training the BP network using various optimizers. As shown in Fig. 39, ARO is the most efficient and its fitness value can reach 0.0469 after 500 iterations. The convergence characteristics of ARO show that the convergence rate is slow in the initial phase, as the iterations increase, the algorithm speeds up the convergence rate and then quickly converges to the optimal fitness in a short period of time. However, several other algorithms have a slower convergence rate and converge to the higher fitness values in the end. This figure suggests that ARO outperforms other algorithms in optimizing the BP network with the feature vectors from the training sample set.

Finally, when the training process is achieved, the feature vectors from the test sample set including three types of faults are fed into

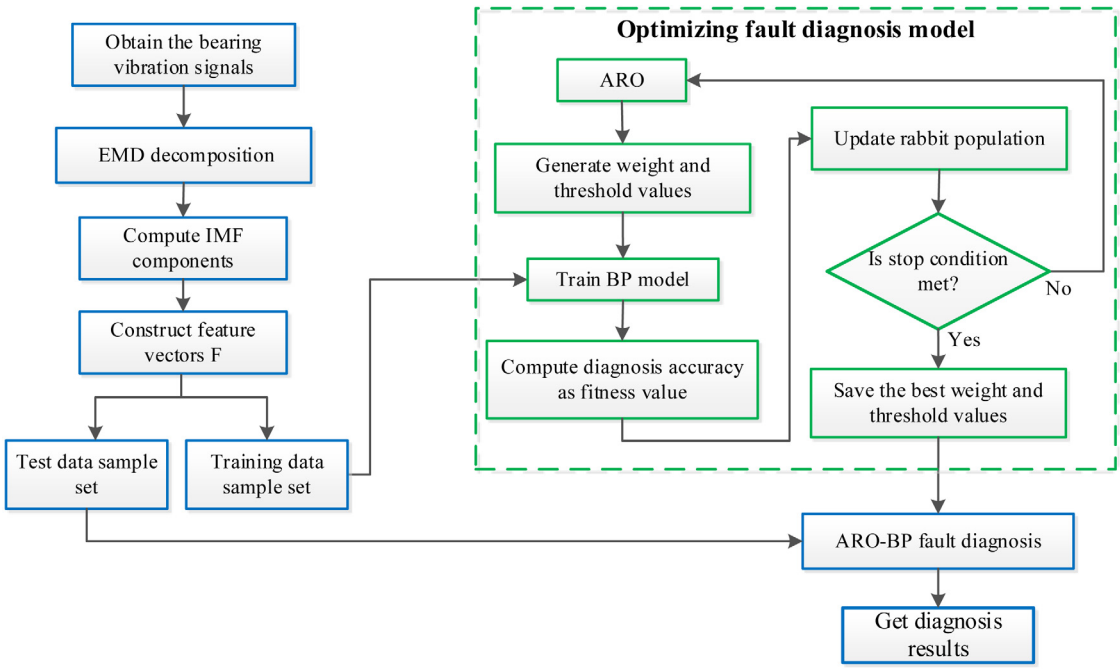


Fig. 36. Model of bearing fault diagnosis based on ARO-BP.

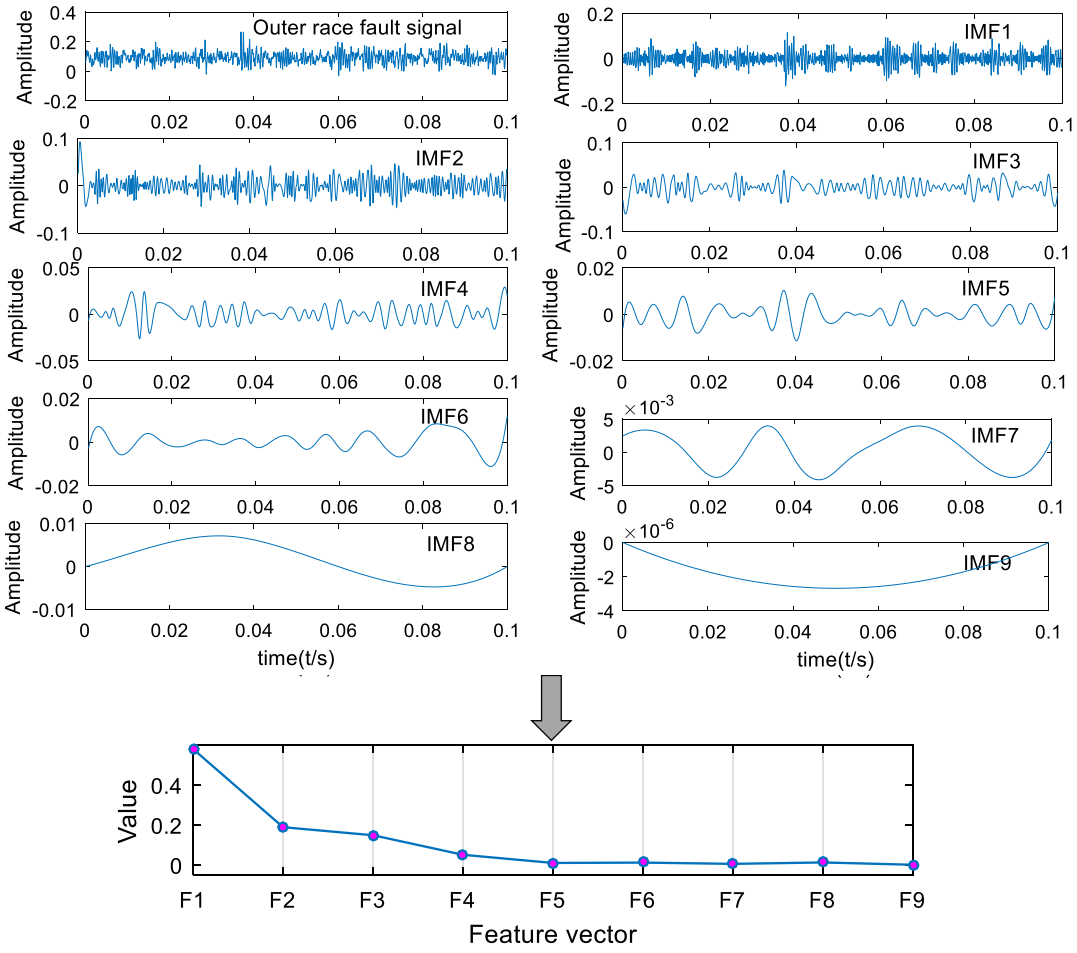


Fig. 37. Extraction process of feature vectors from an original outer race fault signal by EMD decomposition.

the optimized BP network with ARO to conduct the fault diagnosis of rolling bearing. Fig. 40 shows the convergence curve of the training BP network with the gradient descent method during 10000 iterations. Fig. 41 depicts the diagnosis results of the test sample set from the

Table 18
Feature vectors of three types of bearing faults used for BP network.

Sample No.	Fault type	Feature vector (F)									Desired output		
		F1	F2	F3	F4	F5	F6	F7	F8	F9			
1		0.5921	0.1884	0.1230	2.77E−02	4.40E−03	6.95E−03	1.63E−02	4.10E−02	1.20E−07	1	0	0
2	Outer race	0.6616	0.1452	0.1014	4.87E−02	4.17E−03	1.09E−02	1.32E−02	1.47E−02	9.96E−10	1	0	0
3		0.5949	0.2190	0.1050	4.11E−02	1.36E−02	1.21E−02	1.19E−02	2.17E−03	1.36E−04	1	0	0
61		0.7280	0.1811	0.0547	1.95E−02	5.27E−03	4.79E−03	4.00E−03	2.71E−03	4.96E−09	0	1	0
62	Inner race	0.7082	0.2009	0.0526	2.25E−02	7.08E−03	2.28E−03	5.35E−03	1.04E−03	3.06E−09	0	1	0
63		0.7542	0.1456	0.0551	1.86E−02	1.02E−02	1.23E−02	4.05E−03	3.11E−05	9.14E−09	0	1	0
121		0.5840	0.2358	0.1224	3.05E−02	7.08E−03	1.96E−03	5.18E−03	1.16E−02	1.44E−03	0	0	1
122	Roller	0.6132	0.2025	0.0992	6.20E−02	1.12E−02	2.78E−03	7.69E−03	6.78E−04	8.31E−04	0	0	1
123		0.6064	0.1951	0.1053	5.82E−02	1.64E−02	1.09E−02	5.35E−03	2.14E−03	4.94E−05	0	0	1

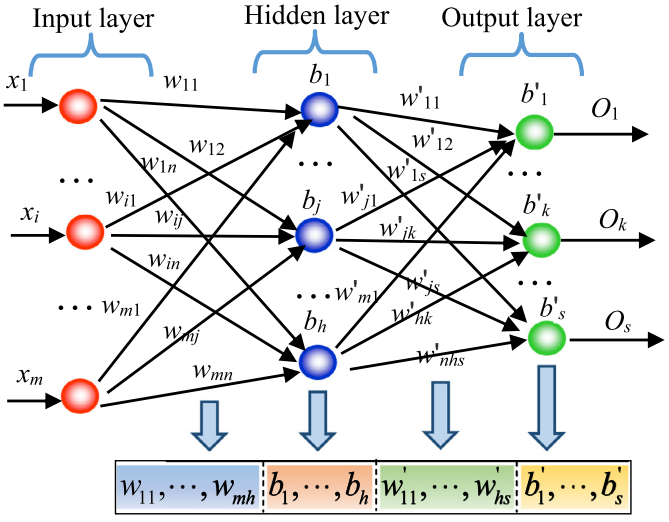


Fig. 38. Architecture of BP network.

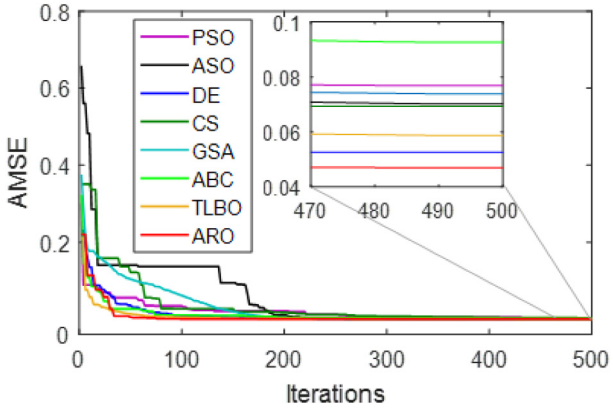


Fig. 39. Convergence curves of training BP network using various optimizers.

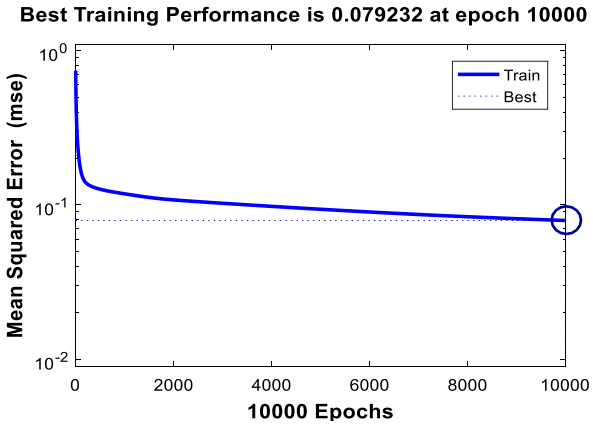


Fig. 40. Convergence curve of training BP network with gradient descent method.

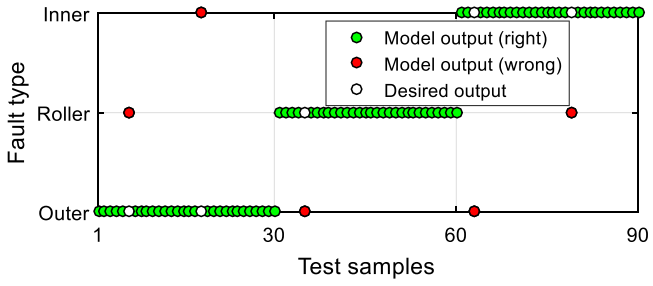


Fig. 41. Diagnosis results of the test sample set using the ARO-BP model.

obtained ARO-BP model. From Fig. 41, two test samples of the outer race fault are misdiagnosed as the inter race fault and roller fault, respectively; two test samples of inner race fault are misdiagnosed as the outer fault and roller fault, respectively; only one test sample of the roller race is misdiagnosed as the outer race fault, the overall diagnostic accuracy of ARO-BP is up to 94.44%.

The diagnosis results of the test sample set with the defeat depth = 0.6 mm using various diagnosis models are illustrated in Fig. 42 and Table 19 lists the comparisons of these diagnostic results. Based on Table 19, it becomes evident that the ARO-BP model has a better average diagnosis precision (94.44%) than the BP network optimized by other optimization algorithms and the GD-BP model. In addition, as shown in Table 19, the diagnosis precision (96.67%) of the inner race fault using the ARO-BP model is the best of all the diagnosis models; the diagnosis precision (96.67%) of the outer race fault using the ARO-BP model is greater than or equal to other diagnosis models; for the roller fault, although the diagnosis precision provided by the ARO-BP model is worse than that of both DE-BP and TLBO-BP models, it is better than the rest models (i.e. PSO-BP, ASO-BP, CS-BP, GSA-BP, ABC-BP, and GD-BP). That is to say, compared to other considered optimization methods, ARO can be better to optimize the weights and threshold of the BP network that contribute to the fault diagnosis, verifying the superiority of ARO. The reason for the excellent diagnosis results of the ARO-BP model is its enhanced exploration and exploitation in the ARO.

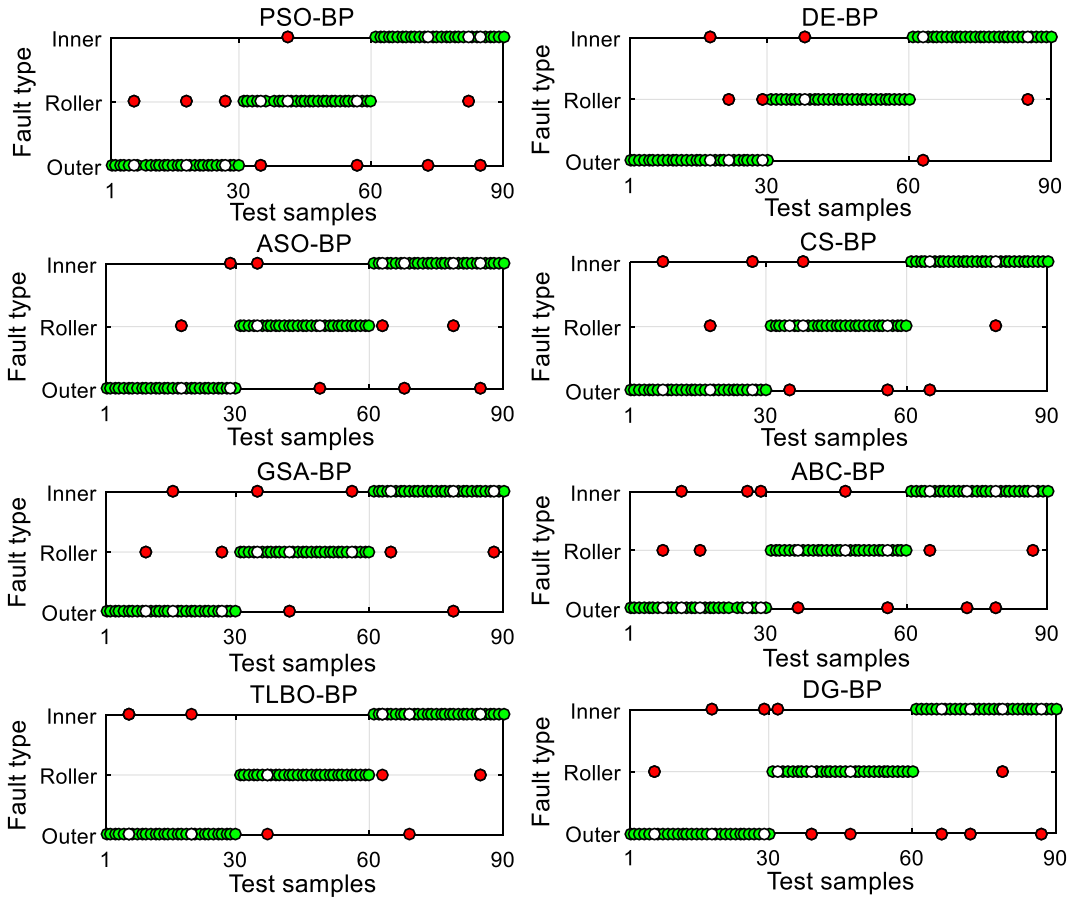


Fig. 42. Diagnosis results of the test sample set various diagnosis models.

In a similar manner, the same experiment process is used in the bearing vibration sample sets with the defeat depth = 0.2 mm and 1 mm. the diagnosis results of different models for three types of vibration sample sets are depicted in Fig. 43. Based on the figure, it can be observed that, although the mean diagnosis accuracy (91.11%) of the ARO-BP model using the defeat depth = 0.2 mm is slightly worse than the mean diagnosis accuracy (92.22%) of TLBO-BP model, it is better than that of the other models. Meanwhile, the diagnosis results from Fig. 43 can clearly show that the mean diagnosis accuracy obtained by the ARO-BP model is the best when using the test sample sets with both the defeat depth=0.6 mm and 1 mm. Additionally, the average of mean diagnosis accuracy of three defeat depths (93.70%) from the ARO-BP model outperforms that from other diagnosis models (i.e. PSO-BP (89.26%), DE-BP (91.85%), ASO-BP (89.26%), CS-BP (88.52%), GSA-BP (88.15%), ABC-BP (85.56%) TLBO-BP (92.96%), and GD-BP (88.89%).

On the other hand, the mean diagnosis accuracy by the majority of diagnosis models increases as the defeat depth of bearing increases. This is probably due to that the more defeat depth in different positions of bearing more highlights the fault features in vibration signals, which is very helpful for fault diagnosis. However, of all considered models, the ARO-BP model is best able to improve the diagnosis accuracy with the increase of defeat depth, indicating that ARO possesses a stable and good search capability for the global optimum.

Therefore, the contrastive analysis of the experiment demonstrates that the performance of ARO for the BP network is superior to other competitive optimizers as demonstrated by the experimental results. Meanwhile, a conclusion can be drawn that ARO is significantly competitive when solving complex engineering optimization problems.

6. Conclusions

This study has developed a new bio-inspired optimizers inspired by rabbits in nature. This proposed algorithm models the survival strategies of rabbits: detour foraging and random hiding. A set of 31 benchmark functions, including unimodal, multimodal and composite functions, are employed to test the performance of ARO. To highlight the optimization capacity, a group of popular meta-heuristics, such as PSO, DE, GSA, CS, TLBO, ABC, and ASO, are used for comparison. In addition, the non-parametric statistical analysis such as the WSRT and Friedman test are also employed to verify the efficiency of the optimizers quantitatively. The comparisons unveil that ARO can determine the global optima for the majority of unimodal, multimodal, and composite functions than its competitors more efficiently. In addition, the results of the semi-real engineering problems reveal that ARO is significantly competitive in tackling engineering tasks with unknown and constrained search space.

Despite good efficacy of ARO in the benchmark functions, a complex engineering problem is considered to display its practicability in tackling the real problems in practice. In this case, the ARO algorithm is employed to tune the BP network for the fault diagnosis of a rolling bearing, the experimental results suggest that a substantial improvement of the diagnosis model tuned by ARO compared to the current methods.

Sum up, the above superior results are attributed to the following several aspects.

- The proposed devour foraging contributes to exploration and assists ARO to perform global search.
- The proposed random hiding is dedicated to exploitation and allows ARO to perform local search.

Table 19
The comparisons of diagnostic results obtained by different diagnosis models.

Diagnosis model	Training accuracy (AMSE)	The testing accuracy obtained using various methods (%)			Average accuracy (%)
		Outer race	Roller	Inner race	
ARO-BP	0.0469	93.33% (28/30)	93.33% (28/30)	96.67% (29/30)	94.44% (85/90)
PSO-BP	0.0769	90.00% (27/30)	90.00% (27/30)	90.00% (27/30)	90.00% (81/90)
DE-BP	0.0525	90.00% (27/30)	96.67% (29/30)	93.33% (28/30)	93.33% (84/90)
ASO-BP	0.0702	93.33% (28/30)	93.33% (28/30)	86.67% (26/30)	91.11% (82/90)
CS-BP	0.0693	90.00% (27/30)	90.00% (27/30)	93.33% (28/30)	91.11% (82/90)
GSA-BP	0.0738	90.00% (27/30)	90.00% (27/30)	90.00% (27/30)	90.00% (81/90)
ABC-BP	0.0926	83.33% (25/30)	90.00% (27/30)	86.67% (26/30)	86.67% (78/90)
TLBO-BP	0.0587	93.33% (28/30)	96.67% (29/30)	90.00% (27/30)	93.33% (84/90)
GD-BP	0.0792	90.00% (27/30)	90.00% (27/30)	86.67% (26/30)	88.89% (80/90)

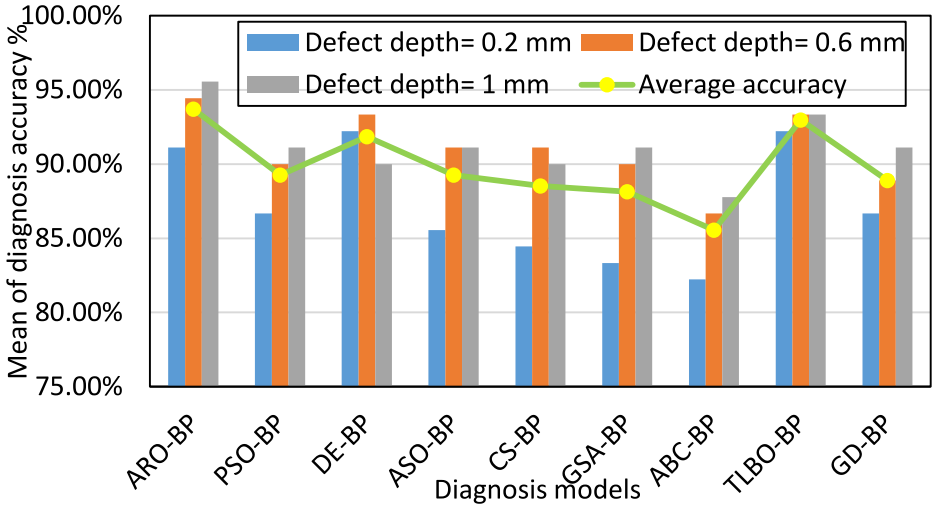


Fig. 43. Diagnostic results of the test sample sets with defect depth = 0.2, 0.6 and 1 mm using various diagnosis models.

- The parameter R can be adjusted adaptively with the increase of iterations, which assists the algorithm to gradually shifts its focus from exploration to exploitation.
- Depending on the value of the energy factor A, half of the iterations are assigned to exploration ($A>1$) and the other half are contributed to exploitation ($A \leq 1$).
- The energy factor A shows a time-dependent dynamic characteristic, which not only effectively enhances both the exploration and exploitation of ARO, but only gradually switches from the exploration phase to the exploitation phase.
- The superior ability of bearing fault diagnosis provided by ARO ascribes to its right balance between exploitative and explorative searches.

Despite the superior performance, ARO has several shortcomings too. The superiority of ARO is not remarkable when handling some unimodal problems with complex shapes and plentiful local extrema. There is a lack of multiple different search mechanisms in exploration and exploitation stages while optimizing the problems with different types of uncertainty. The shortcomings make ARO less powerful than other algorithms in solving NP-hardness problems. The shortcomings, however, open up some promising research directions and potentials. In future works, on the one hand, tuning other types of neural networks and classification frameworks by ARO can be investigated for the fault diagnosis of a rolling bearing or other mechanical devices. On the other hand, the binary and multi-objective versions of ARO can be developed, respectively.

CRediT authorship contribution statement

Liying Wang: Writing – original draft, Methodology, Software, Visualization, Investigation, Funding acquisition, Supervision. **Qingjiao Cao:** Formal analysis, Investigation, Writing – review & editing, Visualization. **Zhenxing Zhang:** Writing – review & editing, Formal analysis, Investigation, Visualization. **Seyedali Mirjalili:** Writing – review & editing, Drawing, Visualization, Investigation. **Weiguo Zhao:** Conceptualization, Software, Visualization, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are very grateful to the reviewers for their valuable and insightful suggestions and comments, which helped us to improve the paper. This work was supported in part by National Natural Science Foundation of China (11972144, 12072098), and One Hundred Outstanding Innovative Scholars of Colleges and Universities in Hebei Province of China (SLRC2019022). The authors gratefully acknowledge the work and help of Prof. Jiqiang Chen.

Appendix

See Tables A.1–A.4.

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	$F_7(x)=\sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	$[-1.28,1.28]^n$	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	$[-500,500]^n$	−12569.5
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
Penalized 2	$F_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{29}(x_i-1)^2 p[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_{30})]\}+\sum_{i=1}^{30}u(x_i,5,10,4)$	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	$F_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1+1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	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	$F_{21}(x)=-\sum_{i=1}^{5}\lvert(x_i-a_i)(x_i-a_i)^T+c_i\rvert^{-1}$	4	$[0, 10]^n$	−10.1532
Shekel 7	$F_{22}(x)=-\sum_{i=1}^{7}\lvert(x_i-a_i)(x_i-a_i)^T+c_i\rvert^{-1}$	4	$[0, 10]^n$	−10.4028
Shekel 10	$F_{23}(x)=-\sum_{i=1}^{10}\lvert(x_i-a_i)(x_i-a_i)^T+c_i\rvert^{-1}$	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

References

Abedinpourshotorban, H., Shamsuddin, S.M., Beheshti, Z., Jawawi, D.N., 2016. Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm. Swarm Evol. Comput. 26, 8–22.

Ai, N., Wu, B., Li, B., Zhao, Z., 2021. 5G heterogeneous network selection and resource allocation optimization based on cuckoo search algorithm. Comput. Commun. 168, 170–177.

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

Alatas, B., 2011. ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst. Appl. 38 (10), 13170–13180.

Barshandeh, S., Haghzadeh, M., 2020. A new hybrid chaotic atom search optimization based on tree-seed algorithm and levy flight for solving optimization problems. Eng. Comput. 1–44.

Barshandeh, S., Piri, F., Sangani, S.R., 2020. HMPA: an innovative hybrid multi-population algorithm based on artificial ecosystem-based and Harris Hawks optimization algorithms for engineering problems. Eng. Comput. 1–45.

Bebis, G., Georgiopoulos, M., 1994. Feed-forward neural networks. IEEE Potent. 13 (4), 27–31.

Beldjilali, B., Benadda, B., Sadouni, Z., 2020. Vehicles circuits optimization by combining GPS/GSM information with metaheuristic algorithms. Romanian J. Inf. Sci. Technol. 23 (T), 5–17.

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

Bohat, V.K., Arya, K.V., 2018. An effective gbest-guided gravitational search algorithm for real-parameter optimization and its application in training of feedforward neural networks. Knowl.-Based Syst. 143, 192–207.
Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Inform. Sci. 237, 82–117.
Camp, M.J., Rachlow, J.L., Shipley, L.A., Johnson, T.R., Bockting, K.D., 2014. Grazing in sagebrush rangelands in western north america: implications for habitat quality for a sagebrush specialist, the pygmy rabbit. Rangel. J. 36 (2), 151–159.
Chen, T., Babanin, A., Muhammad, A., Bert, C., Cyj, C., 2020. Modified evolved bat algorithm of fuzzy optimal control for complex nonlinear systems. Rom. J. Inf. Sci. Tech. 23, T28–T40.
Chen, N., Li, Y., Liu, X., Zhang, Z., 2021. A mutual information based federated learning framework for edge computing networks. Comput. Commun. 176, 23–30.
Cheng, M.Y., Prayogo, D., 2014. Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput. Struct. 139, 98–112.
Chickermane, H., Gea, H.C., 1996. Structural optimization using a new local approximation method. Internat. J. Numer. Methods Engrg. 39 (5), 829–846.
Civicioglu, P., 2012. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. Comput. Geosci. 46, 229–247.
Civicioglu, P., 2013. Backtracking search optimization algorithm for numerical optimization problems. Appl. Math. Comput. 219 (15), 8121–8144.
Coello, C.A.C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput. Methods Appl. Mech. Engrg. 191 (11–12), 1245–1287.
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 (3), 193–203.
Das, B., Mukherjee, V., Das, D., 2020. Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems. Adv. Eng. Softw. 146, 102804.
de Vasconcelos Segundo, E.H., Mariani, V.C., dos Santos Coelho, L., 2019a. Design of heat exchangers using Falcon optimization algorithm. Appl. Therm. Eng. 156, 119–144.
de Vasconcelos Segundo, E.H., Mariani, V.C., dos Santos Coelho, L., 2019b. Metaheuristic inspired on owls behavior applied to heat exchangers design. Therm. Sci. Eng. Progr. 14, 100431.
Deb, K., Goyal, M., 1996. A combined genetic adaptive search (GeneAS) for engineering design. Comput. Sci. Inf. 26, 30–45.
Derrac, J., García, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1 (1), 3–18.
Dhiman, G., Kaur, A., 2019. STOA: a bio-inspired based optimization algorithm for industrial engineering problems. Eng. Appl. Artif. Intell. 82, 148–174.
Dhiman, G., Kumar, V., 2019. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowl.-Based Syst. 165, 169–196.
Drira, N., Kotti, M., Fakhfakh, M., Siarry, P., Tlelo-Cuautle, E., 2020. Convergence rates of the efficient global optimization algorithm for improving the design of analog circuits. Anal. Integr. Circu. Signal Proc. 1–20.
Duan, Y., Liu, C., Li, G., Chunlin, Y., 2021. Manta ray foraging and Gaussian mutation-based elephant herding optimization for global optimization. Eng. Comput. http://dx.doi.org/10.1007/s00366-021-01494-5.
Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. In: MHS'95, Ieee, pp. 39–43.
Ertenlice, O., Kalayci, C.B., 2018. A survey of swarm intelligence for portfolio optimization: Algorithms and applications. Swarm Evol. Comput. 39, 36–52.
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.
Faramarzi, A., Heidarinejad, M., Stephens, B., Mirjalili, S., 2020. Equilibrium optimizer: A novel optimization algorithm. Knowl.-Based Syst. 191, 105190.
Fattahi, E., Bidar, M., Kanan, H.R., 2018. Focus group: an optimization algorithm inspired by human behavior. Int. J. Comput. Intell. Appl. 17 (01), 1850002.
Firestone, C.Z., Warren, W.H., 2010. Why does the rabbit escape the fox on a zig-zag path? Predator-prey dynamics and the constant bearing strategy. J. Vis. 10 (7), 1049.
Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Amer. Statist. Assoc. 32 (200), 675–701.
Gandomi, A.H., Yang, X.S., Alavi, A.H., 2013. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng. Comput. 29 (1), 17–35.
Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001. A new heuristic optimization algorithm: harmony search. Simulation 76 (2), 60–68.
Girjashankar, P.R., Upadhyaya, T., 2021. Substrate integrated waveguide fed dual band quad-elements rectangular dielectric resonator MIMO antenna for millimeter wave 5G wireless communication systems. AEU-Int. J. Electron. Commun. 137, 153821.
Güler, I., Meghdadi, M., 2008. A different approach to off-line handwritten signature verification using the optimal dynamic time warping algorithm. Digit. Signal Process. 18 (6), 940–950.
Hashim, F.A., Houssein, E.H., Mabrouk, M.S., Al-Atabany, W., Mirjalili, S., 2019. Henry gas solubility optimization: A novel physics-based algorithm. Future Gener. Comput. Syst. 101, 646–667.
Hatamlou, A., 2013. Black hole: A new heuristic optimization approach for data clustering. Inform. Sci. 222, 175–184.
Hayyolalam, V., Kazem, A.A.P., 2020. Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Eng. Appl. Artif. Intell. 87, 103249.
He, Q., Wang, L., 2007a. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl. Math. Comput. 186 (2), 1407–1422.
He, Q., Wang, L., 2007b. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. 20 (1), 89–99.
Heaton, J., 2008. Introduction to Neural Networks with Java. Heaton Research, Inc.
Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. 97, 849–872.
Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, USA.
Hosseini, H.S., 2007. Problem solving by intelligent water drops. In: 2007 IEEE Congress on Evolutionary Computation. IEEE, pp. 3226–3231.
Hu, L., Yan, H., Li, L., Pan, Z., Liu, X., Zhang, Z., 2021. MHAT: An efficient model-heterogenous aggregation training scheme for federated learning. Inform. Sci. 560, 493–503.
Huang, F.Z., Wang, L., He, Q., 2007. An effective co-evolutionary differential evolution for constrained optimization. Appl. Math. Comput. 186 (1), 340–356.
Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y., 2018. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. Neural Comput. Appl. 31 (11), 7665–7683.
Israr, A., Yang, Q., Israr, A., 2022. Power consumption analysis of access network in 5G mobile communication infrastructures—An analytical quantification model. Pervasive Mob. Comput. 101544.
Javidy, B., Hatamlou, A., Mirjalili, S., 2015. Ions motion algorithm for solving optimization problems. Appl. Soft Comput. 32, 72–79.
Juan, Q., 2017. Rabbits do not eat grass around the nest. Knowl. Window 13, 39.
Jung, S.H., 2003. Queen-bee evolution for genetic algorithms. Electron. Lett. 39 (6), 575–576.
Junsheng, C., Dejie, Y., Yu, Y., 2006. Research on the intrinsic mode function (IMF) criterion in EMD method. Mech. Syst. Signal Process. 20 (4), 817–824.
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.
Karaboga, D., Akay, B., 2009. A comparative study of artificial bee colony algorithm. Appl. Math. Comput. 214 (1), 108–132.
Kashan, A.H., 2014. League championship algorithm (LCA): An algorithm for global optimization inspired by sport championships. Appl. Soft Comput. 16, 171–200.
Kaveh, A., 2017. Tug of war optimization. In: Advances in Metaheuristic Algorithms for Optimal Design of Structures. Springer, Cham, pp. 451–487.
Kaveh, A., Bakhshpoori, T., 2016. Water evaporation optimization: a novel physically inspired optimization algorithm. Comput. Struct. 167, 69–85.
Kaveh, A., Mahdavi, V.R., 2014. Colliding bodies optimization method for optimum discrete design of truss structures. Comput. Struct. 139, 43–53.
Kaveh, A., Talatahari, S., 2010a. A novel heuristic optimization method: charged system search. Acta Mech. 213 (3), 267–289.
Kaveh, A., Talatahari, S., 2010b. An improved ant colony optimization for constrained engineering design problems. Eng. Comput. 27, 155–182.
Kaveh, A., Talatahari, S., Khodadadi, N., 2020. Stochastic paint optimizer: theory and application in civil engineering. Eng. Comput. 1–32.
Klein, C.E., Mariani, V.C., dos Santos Coelho, L., 2018. Cheetah based optimization algorithm: A novel swarm intelligence paradigm. In: ESANN. Bruges, Belgium.
Kramer, O., 2010. A review of constraint-handling techniques for evolution strategies. Appl. Comput. Intell. Soft Comput. 2010.
Li, L.J., Huang, Z.B., Liu, F., Wu, Q.H., 2007. A heuristic particle swarm optimizer for optimization of pin connected structures. Comput. Struct. 85 (7–8), 340–349.
Li, L.X., Shao, Z.J., Qian, J.X., 2002. An optimizing method based on autonomous animals: fish-swarm algorithm. Syst. Eng. – Theory Prac. 22 (11), 32–38.
Li, M.D., Zhao, H., Weng, X.W., Han, T., 2016. A novel nature-inspired algorithm for optimization: Virus colony search. Adv. Eng. Softw. 92, 65–88.
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. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, p. 635, 490.
Liao, Y., Zhao, W., Wang, L., 2021. Improved Manta ray foraging optimization for parameters identification of magnetorheological dampers. Mathematics 9 (18), 2230.
Liu, M., 2017. Five-elements cycle optimization algorithm for solving continuous optimization problems. In: 2017 IEEE 4th International Conference on Soft Computing & Machine Intelligence. ISCMI, IEEE, pp. 75–79.
Lynn, N., Suganthan, P.N., 2015. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm Evol. Comput. 24, 11–24.
Ma, Z., Zhang, W., Luo, Z., Sun, X., Li, Z., Lin, L., 2020. Ultrasonic characterization of thermal barrier coatings porosity through BP neural network optimizing Gaussian process regression algorithm. Ultrasonics 100, 105981.

Mahdavi, M., Fesanghary, M., Damangir, E., 2007. An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. 188 (2), 1567–1579.
Mezura-Montes, E., Coello, C.A.C., 2005. Useful infeasible solutions in engineering optimization with evolutionary algorithms. In: Mexican International Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, pp. 652–662.
Mezura-Montes, E., Coello, C.A.C., 2008. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int. J. Gen. Syst. 37 (4), 443–473.
Mirjalili, S., 2015. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl.-Based Syst. 89, 228–249.
Mirjalili, S., 2016. SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133.
Mirjalili, S., Gandomi, A.H., 2017. Chaotic gravitational constants for the gravitational search algorithm. Appl. Soft Comput. 53, 407–419.
Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M., 2017. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. 114, 163–191.
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.
Mohammadi, D., Abd Elaziz, M., Moghdani, R., Demir, E., Mirjalili, S., 2021. Quantum Henry gas solubility optimization algorithm for global optimization. Eng. Comput. 1–20.
Moosavi, S.H.S., Bardsiri, V.K., 2019. Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Eng. Appl. Artif. Intell. 86, 165–181.
Mousavirad, S.J., Ebrahimpour-Komleh, H., 2017. Human mental search: a new population-based metaheuristic optimization algorithm. Appl. Intell. 47 (3), 850–887.
Naruei, I., Keynia, F., 2021. Wild horse optimizer: a new meta-heuristic algorithm for solving engineering optimization problems. Eng. Comput. 1–32.
Nematollahi, A.F., Rahiminejad, A., Vahidi, B., 2017. A novel physical based meta-heuristic optimization method known as lightning attachment procedure optimization. Appl. Soft Comput. 59, 596–621.
Ning, A.P., Zhang, X.Y., 2013. Convergence analysis of artificial bee colony algorithm. Control Decis. 28 (10), 1554–1558.
Nowcki, H., 1974. Optimization in pre-contract ship design. In: Fujita, Y., Lind, K., Williams, T.J. (Eds.), Computer Applications in the Automation of Shipyard Operation and Ship Design, Vol. 2. NorthHolland, Elsevier, New York, pp. 327–338.
Park, S., Suh, Y., Lee, J., 2021. FedPSO: federated learning using particle swarm optimization to reduce communication costs. Sensors 21 (2), 600.
Pierezan, J., Coelho, L.D.S., 2018. Coyote optimization algorithm: a new metaheuristic for global optimization problems. In: 2018 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 1–8.
Pierezan, J., Maidl, G., Yamao, E.M., dos Santos Coelho, L., Mariani, V.C., 2019. Cultural coyote optimization algorithm applied to a heavy duty gas turbine operation. Energy Convers. Manage. 199, 111932.
Pierezan, J., dos Santos Coelho, L., Mariani, V.C., de Vasconcelos Segundo, E.H., Prayogo, D., 2021. Chaotic coyote algorithm applied to truss optimization problems. Comput. Struct. 242, 106353.
Połap, D., Woźniak, M., 2017. Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism. Symmetry 9 (10), 203.
Połap, D., Woźniak, M., 2021. Meta-heuristic as manager in federated learning approaches for image processing purposes. Appl. Soft Comput. 113, 107872.
Pozna, C., Precup, R.E., Horvath, E., Petriu, E.M., 2022. Hybrid particle filter-particle swarm optimization algorithm and application to fuzzy controlled servo systems. IEEE Trans. Fuzzy Syst..
Precup, R.E., David, R.C., Roman, R.C., Petriu, E.M., Szedlak-Stinean, A.I., 2021. Slime mould algorithm-based tuning of cost-effective fuzzy controllers for servo systems. Int. J. Comput. Intell. Syst. 14 (1), 1042–1052.
Punnathanam, V., Kotecha, P., 2016. Yin-Yang-pair optimization: A novel lightweight optimization algorithm. Eng. Appl. Artif. Intell. 54, 62–79.
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. Inform. 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. Inform. Sci. 179 (13), 2232–2248.
Rasheed, I., 2022. An effective approach for initial access in 5G-millimeter wave-based vehicle to everything (V2X) communication using improved genetic algorithm. Phys. Commun. 52, 101619.
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.
Reynolds, R.G., 1994. An introduction to cultural algorithms. In: Evolutionary Programming-Proceedings of the Third Annual Conference, 24. World Scientific Press, San Diego, CA, USA, pp. 131–139.
Rodríguez-Barroso, N., Stipcich, G., Jiménez-López, D., Ruiz-Millán, J.A., Martínez-Cámara, E., González-Seco, G., Herrera, F., 2020. Federated learning and differential privacy: Software tools analysis, the Sherpa. ai FL framework and methodological guidelines for preserving data privacy. Inf. Fusion 64, 270–292.
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.
Savsani, P., Savsani, V., 2016. Passing vehicle search (PVS): a novel metaheuristic algorithm. Appl. Math. Model. 40 (5–6), 3951–3978.
Shah-Hosseini, H., 2011. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int. J. Comput. Sci. Eng. 6 (1–2), 132–140.
Shefaei, A., Vahid-Pakdel, M.J., Mohammadi-Ivatloo, B., 2018. Application of a hybrid evolutionary algorithm on reactive power compensation problem of distribution network. Comput. Electr. Eng. 72, 125–136.
Siarry, P. (Ed.), 2016. Metaheuristics. Springer International Publishing.
Solis, F.J., Wets, RJB., 1981. Minimization by random search techniques. Math. Oper. Res. 6 (1), 19–30.
Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359.
Tahani, M., Babayan, N., 2019. Flow regime algorithm (FRA): a physics-based meta-heuristics algorithm. Knowl. Inf. Syst. 60 (2), 1001–1038.
Tůmová, E., Martinec, M., Chodová, D., 2011. Analysis of Czech rabbit genetic resources. Sci. Agric. Bohem. 42, 113–118.
Tynes, V. V. (Ed.), 2010. Behavior of Exotic Pets. John Wiley & Sons.
Vahidi, B., Foroughi Nematolahi, A., 2019. Physical and physic-chemical based optimization methods: a review. J. Soft Comput. Civ. Eng. 3 (4), 12–27.
Van Den Bergh, F., Engelbrecht, A.P., 2006. A study of particle swarm optimization particle trajectories. Inform. Sci. 176 (8), 937–971.
Wei, Z., Huang, C., Wang, X., Han, T., Li, Y., 2019. Nuclear reaction optimization: A novel and powerful physics-based algorithm for global optimization. IEEE Access 7, 66084–66109.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Wu, G., Pedrycz, W., Suganthan, P.N., Mallipeddi, R., 2015. A variable reduction strategy for evolutionary algorithms handling equality constraints. Appl. Soft Comput. 37, 774–786.
Wu, L., Yang, Y., Maheshwari, M., Li, N., 2019. Parameter optimization for FPSO design using an improved FOA and IFOA-BP neural network. Ocean Eng. 175, 50–61.
Xie, L.P., Zeng, J.C., 2010. The performance analysis of artificial physics optimization algorithm driven by different virtual forces. ICIC Express Lett. (ICIC-EL) 4 (1), 239–244.
Yang, X.S., 2010. Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. 2 (2), 78–84.
Yang, X.S., Deb, S., 2009. Cuckoo search via Lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing. NaBIC, Ieee, pp. 210–214.
Yi-Fei, Pu, Siarry, P, Wu-Yang, Zhu, Jian, Wang, Zhang, N, 2022. Fractional-order ant colony algorithm: A fractional long term memory based cooperative learning approach. Swarm Evol. Comput. 69, 101014.
Yu, F., Xu, X., 2014. A short-term load forecasting model of natural gas based on optimized genetic algorithm and improved BP neural network. Appl. Energy 134, 102–113.
Zamfirache, I.A., Precup, R.E., Roman, R.C., Petriu, E.M., 2022. Policy iteration reinforcement learning-based control using a grey Wolf optimizer algorithm. Inform. Sci. 585, 162–175.
Zhang, X., Wang, H., Sun, B., Li, W., Wang, R., 2013. The Markov model of bean optimization algorithm and its convergence analysis. Int. J. Comput. Intell. Syst. 6 (4), 609–615.
Zhao, W., Wang, L., Mirjalili, S., 2022. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. Comput. Methods Appl. Mech. Engrg. 388, 114194.
Zhao, W., Wang, L., Zhang, Z., 2019a. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. Knowl.-Based Syst. 163, 283–304.
Zhao, W., Wang, L., Zhang, Z., 2019b. Supply–demand-based optimization: a novel economics-inspired algorithm for global optimization. IEEE Access 7, 73182–73206.
Zhao, W., Wang, L., Zhang, Z., 2020a. Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural Comput. Appl. 32, 9383–9425.
Zhao, W., Zhang, Z., Wang, L., 2020b. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng. Appl. Artif. Intell. 87, 103300.