Applied Soft Computing Journal 77 (2019) 225–235

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Enhanced grey wolf optimizer with a model for dynamically estimating the location of the prey

Kaiping Luo *

Department of Management Science and Engineering, Beihang University, Beijing 100191, PR China
Key Laboratory of Complex System Analysis, Management and Decision (Beihang University), Ministry of Education, Beijing 100191, PR China

HIGHLIGHTS

- An enhanced grey wolf optimizer is developed.
- The search bias towards the origin of the coordinate system is well settled.
- The main feature of the social hierarchical consideration is highlighted.
- The potential location of the prey (the global optimal solution) is dynamically estimated.

ARTICLE INFO

Article history:
Received 1 September 2018
Received in revised form 13 December 2018
Accepted 18 January 2019
Available online 28 January 2019

Keywords:
Optimization
Meta-heuristics
Swarm intelligence
Grey wolf optimizer

ABSTRACT

Grey Wolf Optimizer (**GWO**) is a new meta-heuristic inspired by the hunting behavior of grey wolves. Our findings reveal that the optimizer has a strong search bias towards the origin of the coordinate system. In this article, a more realistic model is proposed to mimic the leadership hierarchy and group hunting mechanism of grey wolves in nature. In the innovative model, the location of the prey is dynamically estimated by leader wolves and each wolf is directly moving towards the estimated location of the prey. The proposed algorithm is compared with the original grey wolf optimizer and its recent variants on the CEC2017 test suite. The experimental results indicate that the enhanced optimizer significantly outperforms the original version and recent variants in terms of the convergence speed and the quality of solution found. The proposed algorithm also achieves the best solutions in solving two real engineering optimization problems at a lower computation cost.

1. Introduction

Swarm intelligence is an efficient technique for solving numerical optimization problems, especially in handling non-linear, non-convex or discontinuous problems. Many meta-heuristics have been developed by mimicking the social behavior of various creatures like birds, ants, bees, spider and so on. For example, Particle Swarm Optimization [1], Ant Colony Optimization [2], Artificial Bee Colony [3], Firefly Algorithm [4], Cuckoo Search [5], Ant Lion Optimizer [6], Social Spider Algorithm [7], etc.

In this domain, a new algorithm called Grey Wolf Optimizer (**GWO**) [8] simulates the hunting mechanism of the grey wolf pack, including searching prey, tracking, encircling and then attacking. Unlike other intelligent meta-heuristics, the algorithm takes leadership hierarchy into consideration. Due to the distinctive perspective, the GWO algorithm has been increasingly focused and successfully applied to some real engineering problems. Gupta and Saxena [9] used the GWO algorithm to estimate the parameters of the proportional integral controller for a power system. By the GWO algorithm, Chaman [10] designed a superdefect photonic crystal filter with three or five ellipse-shaped rods. Sulaiman et al. [11] used the GWO algorithm to tackle a reactive power dispatch problem, which is a well-known nonlinear optimization problem in power system. El-Fergany et al. [12] hybridized the grey wolf optimizer and the differential evolution algorithm to solve the optimal power flow problem. Song et al. [13] proposed a new surface wave dispersion curve inversion scheme based on the GWO algorithm. Komaki and Kayvanfar [14] addressed the two-stage assembly flow shop scheduling problem with release time in many industrial areas, such as computer manufacturing, fire engine plant, etc. Mirjalili [15] trained a multi-layer perceptron using the GWO algorithm. Emary et al. [16] proposed a binary grey wolf optimization algorithm and then applied to an optimal feature subset selection problem for the purpose of accurate classification. Medjahed et al. [17] proposed a GWO-based approach to reducing the dimensionality of hyperspectral images. To solve a real scheduling problem in welding production, Lu et al. [18] developed

* Correspondence to: Department of Management Science and Engineering, Beihang University, Beijing 100191, PR China.
E-mail address: kaipingluo@buaa.edu.cn.

https://doi.org/10.1016/j.asoc.2019.01.025
1568-4946/

a multi-objective discrete grey wolf optimizer considering production efficiency and machine load. Zhang et al. [19] used the GWO algorithm to design the unmanned combat aerial vehicle path in the two-dimension coordinate. Jayabarathi et al. [20] introduced the crossover and mutation operations of the genetic algorithm into the GWO algorithm for addressing the economic dispatch problem, which is nonlinear, non-convex, discontinuous, and with numerous equality and inequality constraints. Tawhid et al. [21] also hybridized the grey wolf optimizer and the genetic algorithm to minimize the energy function of the molecule. Kumar et al. [22] utilized the GWO algorithm to solve the reliability optimization problem in the complex system. Precup et al. [23] proposed a tuning approach for fuzzy control systems with a reduced parametric sensitivity using the GWO algorithm. Khairuzzaman et al. [24] developed a GWO-based approach to multilevel thresholding selection in image segmentation. By a fixed-size external archive, Mirjalili et al. [25] expanded the optimizer into the multi-objective case. Sahoo et al. [26] used the multi-objective GWO algorithm to select optimal feature subset for cervix lesion classification.

However, Long et al. [27] pointed out that the GWO algorithm is prone to stagnation at the local optima in solving complex multi-modal problems. Therefore, there are several variants to improve its performances. Saremi et al. [28] employed the strategy of the survival of the fittest in evolutionary algorithms to improve the median fitness of population. More precisely, the worst wolf is repositioned around the leader wolves or in the whole search ranges. To add the chance of exploration, Mitta et al. [29] suggested to use a quadratic formula instead of the linear decreasing coefficient. Heidari and Pahlavani [30] employed the Levy flight and greedy selection strategy. Although the variant [30] can boost the efficacy of the GWO algorithm escaping the local optima in solving many multi-modal problems, the time complexity of the variant becomes larger. Consequently, the simplicity of the optimizer is lost in the variant. Luis et al. [31] embedded a fuzzy hierarchical operator in the optimizer to highlight the leadership hierarchy of the grey wolf pack. Gupta and Deep [32] emphasized the role of the leader wolves and separately implemented a random walk operator for each leader wolf.

The experimental observations discovery that the GWO algorithm has an astonished performance in solving the problem whose global optimal solution is located at the origin of the coordinate system. But, such a miracle will not again emerge once the optimal solution of the problem to be solved is shifted away from the origin of the coordinate system even though the offset is much tiny. In other words, the original grey wolf optimizer has a strong search bias towards the origin of the coordinate system.

To the best of my knowledge, there is no report on the search bias of meta-heuristics towards the origin of the coordinate system. Although Mohsen et al. [33] discovered the center-seeking bias of the gravitational search algorithm [34], our experimental results indicate that the original grey wolf optimizer are more prone to converge into the origin of the coordinate system rather than the center of the search space.

In this paper, a more realistic model is proposed to enhance the efficacy of the grey wolf optimizer, especially in solving the problems whose global optimal solutions are not located at the origin of the coordinate system. The novel model more accurately simulates the hunting mechanism of grey wolves in nature and more vividly characterizes their leadership hierarchy. Compared with the original version [8], the important feature of the proposed model is that the position of the prey is dynamically estimated and the wolf pack moves directly towards the estimated location of the prey. Moreover, the simplicity of the original version is still inherited and the search bias towards the origin of the coordinate system is well settled.

The remainder of this paper is organized as follows. The next section provides a brief introduction of the grey wolf optimizer and investigates its search bias towards the origin of the coordinate system. In Section 3, we propose the enhanced grey wolf optimizer and point out the differences with the original grey wolf optimizer and similar operations in other evolutionary algorithms. The computational complexity of the proposed algorithm is also analyzed in this section. In Section 4, we investigate parameter selection for the enhanced grey wolf optimizer and then test whether or not it has the same search bias as the original grey wolf optimizer. In Section 5, we report and discuss the comparative experimental results of the proposed algorithm and other related competitive algorithms. Section 6 presents results of application to real engineering optimization problems. The last section highlights conclusions and provides suggestions for future research.

2. Search bias of the original grey wolf optimizer

2.1. Grey wolf optimizer

The grey wolf optimizer [8] is a new meta-heuristic inspired by the hunting behavior of the grey wolf pack, including searching prey, tracking, encircling and then attacking. The main feature of the algorithm is the consideration of the social leadership hierarchy. The grey wolf pack has a very strict social dominant hierarchy which is classified into four levels. The wolf in the top of the hierarchy pyramid called alpha α is responsible for making decision about hunting. The wolf in the second level called beta β helps the alpha in decision-making or other pack activities. The beta reinforces the alpha's commands throughout the pack and provides feedback to the alpha. The lowest ranking wolf is omega ω. The omega plays the role of scapegoat. Other wolves such as scouts and sentinels are called delta δ. The delta has to submit to the alpha and the beta, but dominate the omega.

In the GWO algorithm,

(1) the number of search agents is the size of the grey wolf pack;
(2) a solution represents the position of a grey wolf;
(3) the quality of a solution reflects the score of the position of the corresponding wolf;
(4) the global optimal solution implies the location of the prey;
(5) the best three solutions found so far respectively represent the alpha, the beta and the delta in the leadership hierarchy of the wolf pack;
(6) the iteration of the algorithm mimics the hunting process of the wolf pack.

To simulate the hunting behavior guided by the leaders of grey wolves, the GWO algorithm saves the first three best distinct solutions $(x_\alpha, x_\beta, x_\delta)$ found so far and updates every wolf's position by a group of evolving equations as follows.

$$y1 = x_\alpha^j - a1 * |c1 * x_\alpha^j - x_i^j(t)| \tag{1}$$

$$y2 = x_\beta^j - a2 * |c2 * x_\beta^j - x_i^j(t)| \tag{2}$$

$$y3 = x_\delta^j - a3 * |c3 * x_\delta^j - x_i^j(t)| \tag{3}$$

$$x_i^j(t+1) = \frac{y1 + y2 + y3}{3} \tag{4}$$

where $x_i^j(t)$ is the value of the ith solution in the jth dimension at the tth iteration; random numbers $c1$, $c2$ and $c3$ are uniformly distributed in [0,2]; random numbers $a1$, $a2$ and $a3$ are uniformly distributed in $[-2*(1-t/G), 2*(1-t/G)]$; $i = 1, 2, \ldots, m$, indexes each solution in population with size m; $j = 1, 2, \ldots, n$, indexes each dimension of the n-dimensional problem to be solved; $t = 1, 2, \ldots, G$, is the number of iteration.

The fluctuation range of random numbers $a1$, $a2$ and $a3$ is linearly decreasing as the iteration number t gradually increases to a

predetermined maximal number G. To some extent, the decreasing fluctuation characterizes the phenomenon of approaching the prey in the hunting process. If the amplitude is less than 1, the wolf will attack towards the prey and the corresponding search agent will exploit a local optimal solution. If the amplitude is more than 1, the wolf will search for prey and the GWO algorithm will explore more distinct solutions.

The procedure of the GWO algorithm is considerably concise. It starts from a random wolf pack and continually updates the position of each wolf using the above equations until the iterator is beyond a given maximal iteration number or other stopping criterion is satisfied.

Initialization
repeat
  Update the position of each wolf by Eqs. (1)–(4)
  Evaluate the new positions of all wolves
  Update the alpha, the beta, the delta
until the stopping criterion is reached.

2.2. Search bias towards the origin of the coordinate system

Surprisingly, our observations indicate that the GWO algorithm has a super efficacy in solving the problem whose global optimal solution is located at the origin of the coordinate system whether the search space is symmetrical or not. But this miracle will not again emerge once the global optimal solution of the problem to be solved is shifted away from the origin of the coordinate system even though the offset is much tiny.

To verify the drawback, the GWO algorithm solved three well-known representative benchmark problems with 30 dimensions and their variants whose global optimal point are shifted by a small offset. As illustrated in Fig. 1, the Sphere function is unimodal; the Schwefel's Problem 1.2 has a larger stagnation region; the Rastrigin function is multi-modal. The former two functions and their shifted variants were searched in the asymmetrical space while both the later function and the corresponding shifted variant used the symmetrical search space whose center is just the global optimal point of the test function. The GWO algorithm solved each problem using 30 search agents until 1000 iterations are reached. This experiment was repeatedly run over 30 times.

The mathematical expressions of test functions and their search spaces are respectively listed in the first column and in the second column in Table 1. Table 1 also exhibits the average error of each problem and the corresponding standard deviation in the last two columns, respectively. The differences of the performances of the GWO algorithm are dramatic when it solves the original functions and the corresponding shifted variants with a tiny offset, no matter which type the test function is, whether the search space is symmetrical or not. For example, the average error of the Sphere function dramatically grows from 10^{-60} to 10^{-8} after the global optimal point of the test function is shifted to $o_i = 10^{-4}$, $(i = 1, 2, \ldots, 30)$. The average error of the Rastrigin function is also raised from 0.215 to 27.4 after the test function is shifted by a unit offset even though the search space is correspondingly moved by the same offset.

To test the shifting effect on the performances of the original GWO algorithm, a Wilcoxon signed rank test was performed. The last column in Table 2 reports the non-parametric test results. All p-values are much less than the 5% significance level. Namely, the performance of the original GWO algorithm on the three representative functions has been significantly effected by their shifting.

To reduce the family wise error rate [35], the true statistical significance can be calculated through combining pairwise comparisons as follows.

$$
\begin{aligned}
p &= P(\text{reject H0} \mid \text{H0 is true}) \\
&= 1 - P(\text{accept } e_a = e_a^s;\ e_b = e_b^s;\ e_c = e_c^s \mid \text{H0 is true}) \\
&= 1 - P(\text{accept } e_a = e_a^s \mid \text{H0 is true}) \\
&\quad * P(\text{accept } e_b = e_b^s \mid \text{H0 is true}) * P(\text{accept } e_c = e_c^s \mid \text{H0 is true}) \\
&= 1 - \left(1 - 1.73E - 06\right)^3 = 5.20E - 06.
\end{aligned}
$$

The comprehensive p value is much less than the 5% significance level. Thus, we must reject the null hypothesis (H0) that the average errors are the same whether these benchmark functions are shifted or not. In other words, the shifting of the test function has a significant effect on the performances of the original GWO algorithm.

Fig. 2 depicts the process of approaching the global optimal solution of each problem. In Fig. 2, the horizontal axis is the iteration number; the vertical axis is the Euclidian distance of the best solution found to the known global optimal point or to the origin of the coordinate system. The red solid line illustrates that the best solution found by the GWO algorithm quickly approaches the global optimal point of each original test function. The blue dotted line and the green dashed line respectively depict the Euclidean distance of the best solution found to the origin of the coordinate system and to the corresponding shifted optimal point. From this figure, we can see that the speed of the GWO algorithm approaching the original global optimal solution (red solid lines) is remarkably faster than that of approaching the shifted global optimal solution (green dashed lines) even though the offset is much tiny.

It should be noted that the tree sub-figures in Fig. 2 are not the common fitness-based convergence graphs. As shown in the right sub-figure, the three curves are not monotonously decreasing. The distance becomes sometimes longer. Such a case implies that the GWO algorithm successfully jumps over some local optima. If the distance keeps a positive constant, the GWO algorithm is completely trapped into a local optimal point, as shown in the latter parts of the blue line. From the right sub-figure, we can see that the GWO algorithm is more prone to converge towards the local optimal point located at the origin of the coordinate system rather than the near global optimal solution in solving the shifted multi-modal Rastrigin function with a unit offset.

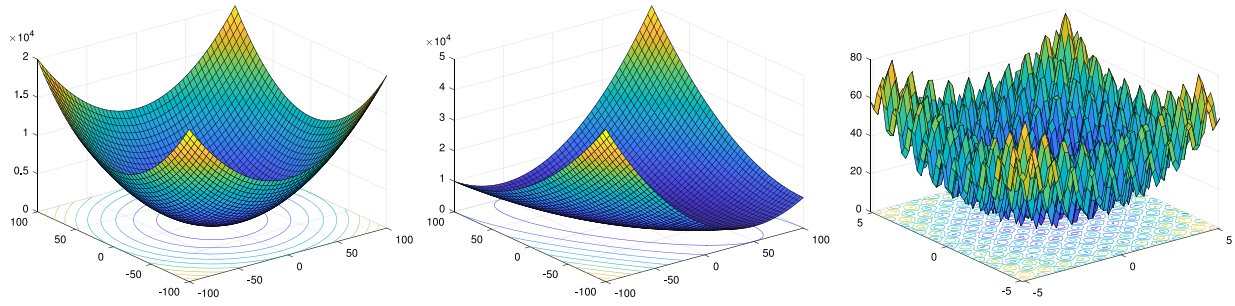
The above observations enough reveal that the original grey wolf optimizer has a strong search bias towards the origin of the coordinate system in solving those problems that have a local optimal solution located at the origin of the coordinate system. Clearly, the bias of the GWO algorithm is not related with the symmetry of the search range. Consequently, this bias is different with the center-seeking bias observed by Mohsen et al. [33] in the gravitational search algorithm.

The shortage of the GWO algorithm should be attributed to its simulated model. If the problem to be solved has a local optimal solution located at the origin of the coordinate system, by Eqs. (1)–(4), the value of $x(t+1)$ will be easily close to 0 and even speedily approach to 0 once $\|x_\alpha\| < 1$, $\|x_\beta\| < 1$ and $\|x_\delta\| < 1$, as we saw in Fig. 2.

In essence, each wolf in the original GWO algorithm practically moves towards the alpha, the beta, the delta rather than the so-called prey. Moreover, the average in Eq. (4) does not seem to reflect the strict leadership hierarchy in the grey wolf pack, as Luis et al. [31] pointed out. Since there is a supposition that these leader wolves have better knowledge about the potential location of prey in the GWO algorithm [8], why do not we estimate the location of the prey based on these leaders' knowledge and then command the wolf pack to proceed directly towards the estimated location of the prey? This paper presents a successful effort.

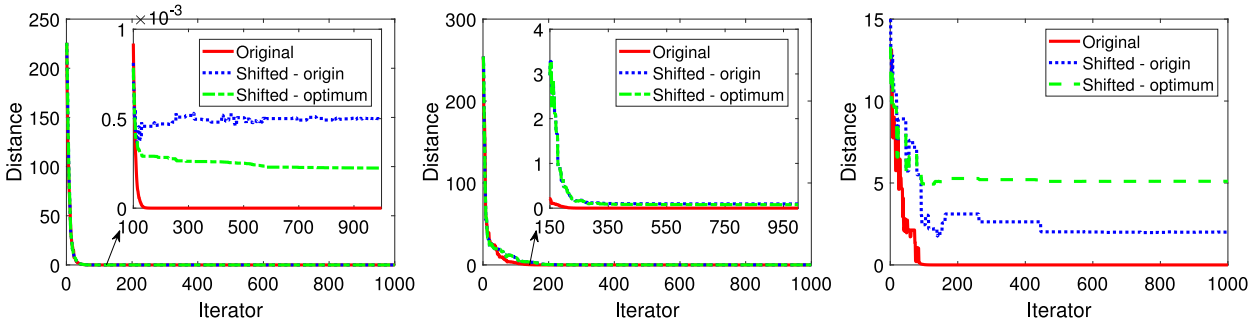
Table 1
Average errors and standard deviations yielded by the GWO algorithm in solving three representative benchmark functions with 30 dimensions and their shifted variants over 30 independent runs.

Function	Range	Mean error	Mean Std.
$f_a(x) = \sum_{i=1}^{n} x_i^2$	[−10,100]	4.75E−60	6.37E−60
$f_a^s(x) = \sum_{i=1}^{n} (x_i - 0.0001)$	[−10,100]	3.63E−08	1.50E−08
$f_b(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_i\right)^2$	[−100,10]	1.15E−16	3.09E−16
$f_b^s(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} (x_i - 0.01)\right)^2$	[−100,10]	2.00E−03	7.00E−04
$f_c(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	[−5.12,5.12]	2.15E−01	1.18E+00
$f_c^s(x) = \sum_{i=1}^{n} \left((x_i - 1)^2 - 10\cos(2\pi(x_i - 1)) + 10\right)$	[−4.12,6.12]	2.74E+01	1.91E+00



(a) f_a: Sphere function (b) f_b: Schwefel's Problem 1.2 (c) f_c: Rastrigin function

Fig. 1. Landscape of the selected representative problems.



(a) Sphere function (b) Schwefel's Problem 1.2 (c) Rastrigin function

Fig. 2. Process of approaching the optimal solution.

3. Enhanced grey wolf optimizer

3.1. Improved model

The main inspiration of the GWO algorithm is derived from the strict social leadership hierarchy of the grey wolf pack and their group hunting behavior. The α wolf in the top of the hierarchy pyramid, which is represented by the best solution found so far, is the decision-maker of pack activities including hunting. In the medium rank, the β wolf and the δ wolf, which are respectively represented by the second and third best solutions found so far, support the α wolf to make decision. The group hunt of the grey wolf pack is usually guided by the α wolf. Both the β wolf and the δ wolf might also participate in the hunting guide occasionally. The GWO algorithm supposes that wolves α, β and δ have better knowledge about the potential location of the prey which is represented by the optimal solution. The supposition is still adopted in this paper. However, the location of the prey is dynamically estimated by the following weight-based formula.

$$x_p^j(t) = w_\alpha * x_\alpha^j(t) + w_\beta * x_\beta^j(t) + w_\delta * x_\delta^j(t) + \epsilon(t) \tag{5}$$

where, $j = 1, 2, \ldots, n$, indexes each dimension of the n-dimensional problem to be solved; $t = 1, 2, \ldots, G$, records the number of iteration up to the given maximal number G; $x_p^j(t)$ denotes the jth component of the estimated location of the prey at the tth iteration; $x_\alpha^j(t)$, $x_\beta^j(t)$ and $x_\delta^j(t)$ are respectively the jth component of the position of wolves α, β and δ at the tth iteration; w_α, w_β and w_δ are respectively the weight of wolves α, β and δ in making the hunting decision. The bigger the value of the weight is, the more dominant right the corresponding wolf has, and vice versa. Since the α wolf has the greatest power in decision-making and the δ wolf the least right, the strict social leadership hierarchy of the grey wolf pack can characterized by the distinct weight $\omega = (w_\alpha, w_\beta, w_\delta)$ that satisfies the following conditions:

$$1 \geqslant w_\alpha > w_\beta > w_\delta \geqslant 0 \tag{6}$$

and

$$w_\alpha + w_\beta + w_\delta = 1. \tag{7}$$

In Eq. (5), $\epsilon(t) \sim N(0, \sigma(t))$ is a simulated stochastic error that follows the Gaussian distribution with mean 0 and standard deviation $\sigma(t)$. The dynamic standard deviation has a property:

$$\sigma(t) > \sigma(t+1). \tag{8}$$

Table 2
Wilcoxon signed rank test of the function shifting effect on the performances of the grey wolf optimizer.

Function	Mean error - Original	Mean error - Shifted	*P*-value
Sphere	4.75E−60	3.63E−08	1.73E−06
Schwdfel 1.2	1.15E−16	2.00E−03	1.73E−06
Rastrign	2.15E−01	2.74E+01	1.73E−06

The above equation mimics the metaphor that the estimation of the leader wolves for the location of the prey becomes more and more precise in the hunting process of the grey wolf pack.

Guided by these leader wolves, each wolf can directly move towards the estimated location of the prey. As a consequence, the position of each wolf i will be updated by the following formula.

$$x_i^j(t+1) = x_p^j(t) - r * |x_p^j(t) - x_i^j(t)| \tag{9}$$

where, $x_i^j(t)$ is the value of the ith solution in the jth dimension at the tth iteration; r is a rand number that uniformly distributed in $[-2, 2]$. If $|r| > 1$, the proposed algorithm will explore new local optima; otherwise, it will continue to exploit a local optimal solution around the estimated location of the prey. As illustrated in the original version of the optimizer [8], the right hand in Eq. (9) mathematically models the encircling behavior of each wolf i; $|r| < 1$ implies attacking prey and $|r| > 1$ searching for prey.

It should be noted that the fluctuation range of random number r is not linearly decreasing from 2 to 0 over the course of iterations. Otherwise, the algorithm will continuously exploit around a local optimal solution in the latter half of iterations because $|r| \leqslant 2*\left(1-\frac{t}{G}\right) < 1$ if $t > \frac{G}{2}$. The fixed fluctuation range, which is different with the decreasing setting in the original GWO algorithm, guarantees the exploratory capability of the optimizer in the latter half of iterations and boosts the efficacy of the optimizer in solving the multi-modal problems, as we will see in the fifth section.

In addition, the hunting operation exhibited in Eq. (9) is different with the 'DE/rand/1' operation in Differential Evolution [36] and the foraging operation in Artificial Bee Colony [3]. Compared with the 'DE/rand/1' operation, the proposed operation does not formally require three distinct individuals in population. In contrast with the foraging operation, the proposed operation generates a neighbor solution over all dimensions rather than only in a randomly-selected dimension. Moreover, the essential difference is that the base $x_p(t)$ in the proposed operation is a dynamically integrated point rather than an existent solution in the current population.

If the new position yielded by (9) is beyond the search bounds, the proposed algorithm will correct it by an arbitrary move towards the bound exceeded while the original GWO algorithm replaces it by the bound. More precisely,

$$x_i^j(t+1) = \begin{cases} x_i^j(t) + u * \left(U^j - x_i^j(t)\right), & \text{if } x_i^j(t+1) > U^j \\ x_i^j(t) + u * \left(L^j - x_i^j(t)\right), & \text{if } x_i^j(t+1) < L^j \end{cases} \tag{10}$$

where U^j and L^j are respectively the jth component of the upper bound and that of the lower bound; u is a random number in [0, 1]. To some extent, the arbitrary move can characterize the random walk behavior of a wolf in the process of searching for a prey.

Compared with the model in the original GWO algorithm [8], the improved model is more realistic. As the main feature of the grey wolf optimizer in the domain of nature-inspired intelligent meta-heuristics, the strict social leadership hierarchy of the grey wolf pack is vividly characterized by the distinct weight (6) in the proposed model. As the important metaphor of the grey wolf optimizer, the supposition that the leader wolves have the better knowledge about the potential location of prey, is well displayed by the weight-based integrated expression (5) with a dynamically simulated error in the proposed model. In addition, the proposed model also considers the random walk behavior of the grey wolf pack, except for searching, encircling and attacking in hunting focused by the original GWO algorithm.

3.2. Whole framework and computational complexity

Base on the above model, we develop an enhanced grey wolf optimizer (**EGWO**). The EGWO algorithm firstly initializes the related parameters and then repeats the proposed hunting operations until the given maximal iteration number is reached or other stopping criterion is satisfied. The proposed algorithm directly estimates the potential location of the global optimal solution by Eq. (5) and then invokes each agent to search around the estimated optimal point. The pseudocode of the enhanced grey wolf optimizer is presented in Algorithm 1.

Algorithm 1 EGWO: enhanced grey wolf optimizer

Input: $m, n, L, U, f(\boldsymbol{x}), G$.

```
Initialize the positions of the wolf pack.
Evaluate the position of each wolf using f(x) and find the leaders: α (the best solution), β (the second best solution) and δ (the third best solution).
for each iteration t = 1 to G do
    Estimate the location of prey using Equation (5).
    for each wolf i = 1 to m do
        for each dimension j = 1 to n do
            Update the position of each wolf i using Equation (9).
            if Exceed the upper bound U or the lower bound L in some dimension then
                Revise it by a random walk towards the bound (10).
            end if
        end for
        Evaluate the new position of each wolf i using f(x).
        Update the leaders.
    end for
end for
```

Output: $f^* = f_\alpha, \boldsymbol{x}^* = \boldsymbol{x}_\alpha$.

According to the pseudocode, the EGWO algorithm has the computational complexity of $O(m * n * G)$, where m is the size of the wolf pack, n the dimension of the problem to be solved and G the maximum number of iteration. In the initialization step, the algorithm has the complexity of $O(m * n)$. In each iteration, the algorithm still requires $O(m * n)$ computational efforts to update three best solutions. After the iterator is up to the given maximum number G, the algorithm will take the computational time of $O(m * n * G)$ in total. Hence, the computational complexity of the EGWO algorithm is identical with that of the original GWO algorithm.

4. Parameter selection and search bias test

4.1. Parameter selection

The enhanced grey wolf optimizer has two control parameters: ω and $\sigma(t)$. In general, the parameters of an algorithm are often problem-dependence. Thus, we focus on the ways to set the two parameters rather than their fine-tuning values in order to guide the users setting.

There are three ways to set the ω weight. One way is to give a fixed assignment by users. For example, $\omega = (0.5, 0.3, 0.2)$ in [31]. Another way is to generate a random assignment at each iteration,

which *must* satisfy conditions (6) and (7). The last way is based on the fitness. More precisely, if the fitness function f is minimized,

$$w_\alpha = 0.5 * \left(1 - \frac{f_\alpha}{f_\alpha + f_\beta + f_\delta}\right),$$
$$w_\beta = 0.5 * \left(1 - \frac{f_\beta}{f_\alpha + f_\beta + f_\delta}\right), \quad (11)$$
$$w_\delta = 0.5 * \left(1 - \frac{f_\delta}{f_\alpha + f_\beta + f_\delta}\right),$$

otherwise,

$$w_\alpha = \frac{f_\alpha}{f_\alpha + f_\beta + f_\delta},\ w_\beta = \frac{f_\beta}{f_\alpha + f_\beta + f_\delta},\ w_\delta = \frac{f_\delta}{f_\alpha + f_\beta + f_\delta}. \quad (12)$$

The $\sigma(t)$ parameter can be set by a linearly or nonlinearly decreasing formula with respect to $t \in \{1, 2, \ldots, G\}$, such as

$$\sigma(t) = 1 - \frac{t}{G} \quad (13)$$

$$\sigma(t) = 1 - \left(\frac{t}{G}\right)^2 \quad (14)$$

$$\sigma(t) = \exp\left(-100 * \frac{t}{G}\right). \quad (15)$$

To investigate the efficacies of the above parameter setting ways, all combinations of these ways are tested on the aforementioned representative benchmark problems with 30 dimensions. The bar chart 3 exhibits the experimental results. In Fig. 3, the x-axis and the y-axis respectively stands for the setting ways for the ω weight and the $\sigma(t)$ parameter; the z-axis is the average of errors under the different settings for the two parameters. As shown in this figure, the exponential formula (15) produces the smallest average error while the quadratic formula (14) yields the biggest average error for the Sphere function and the Schewefel's problem 1.2 regardless of being shifted; the cooperation of the exponential $\sigma(t)$ and the random ω can obtain a minimal average error in solving the multi-modal Rastrigin function. The $\sigma(t)$ parameter seems to have a significant effect on solving unimodal functions while the ω parameter seems to impact on solving multi-modal functions. Thus, the exponentially descending $\sigma(t)$ and the random ω may be suitable for various numerical optimization problems.

4.2. Search bias test

Based on the parameter selection suggested, the enhanced grey wolf optimizer solves the aforementioned representative benchmark problems to test whether or not the proposed algorithm has the same search bias as the original grey wolf optimizer. The test used the same settings in investigating the search bias of the original grey wolf optimizer, i.e., 30 search agents and 1000 iterations. In the test, each problem is repeatedly solved over 30 times. The shifting of the global optimal point is taken as the test factor. Table 3 reports the Wilcoxon signed rank test results. In the table, the middle two columns respectively exhibit the average errors of each original benchmark function and its shifted variant; the last column presents the corresponding P-value. As seen from the table, all P-values are much more than the 5% significance level. The comprehensive P-value, $p = P(\text{reject H0} \mid \text{H0 is true}) = 1 - (1-0.8290)*(1-0.5577)*(1-0.36) = 0.9516 > 0.5$. Therefore, we cannot reject the null hypothesis (H0) that the average error of the original benchmark function is equal to that of the shifted variant. In other words, the test results cannot statistically verify that the enhanced grey wolf optimizer has the search bias towards the origin of the coordinate system.

Table 3
Wilcoxon signed rank test of the function shifting effect on the performances of the enhanced grey wolf optimizer.

Function	Mean error - Original	Mean error - Shifted	P-value
Sphere	4.74E−09	3.79E−09	0.8290
Schwdfel 1.2	2.38E+00	2.04E+00	0.5577
Rastrign	3.41E+01	2.95E+01	0.3600

5. Comparative experiment

5.1. Experimental environment

In the comparative experiment, the CEC2017 test suite [37] is employed. The complex test suite is composed of 3 unimodal functions, 7 simple multi-modal functions, 10 hybrid functions and 10 composition functions. In the suite, all functions are randomly shifted and top 10 functions are rotated at the same time.

The EGWO algorithm is compared with the original GWO algorithm and two recent variants: FH-GWO [31] and RW-GWO [32]. The FH-GWO algorithm uses a fuzzy weight to highlight the hierarchical feature of the grey wolf optimizer while the RW-GWO algorithm specially implements a random walk operation for each leader wolves to boost the local exploitation capability of the optimizer. The proposed algorithm also reflects these ideas: characterizing the strict social leadership hierarchy of the grey wolf pack by the distinct weight and simulating the rank walk behavior of the grey wolf pack. But, there are many differences as explained in Section 3.1.

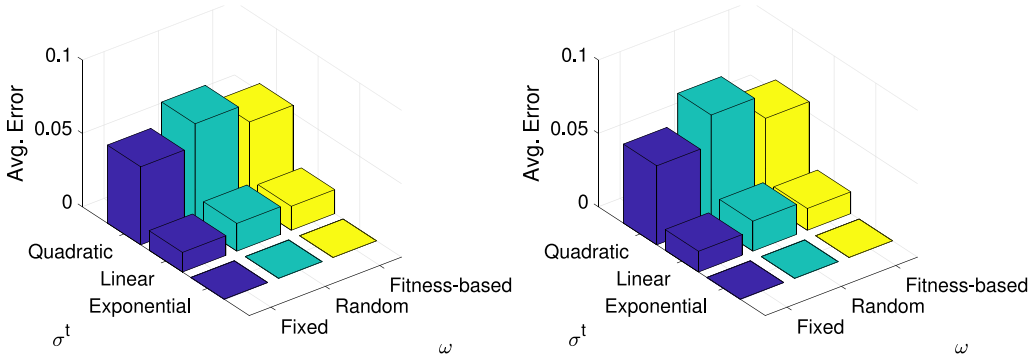
For the sake of fairness, all comparative algorithms stop after the maximal number of function evaluation, NFE $= 10\,000 * n$, is reached. Other original parameter settings for competing algorithms were adopted in the experiment. Each comparative group was independently run over 30 times on a computer with eight 3.0-GHz CPUs and 64 GB of RAM.

5.2. Results and discussion

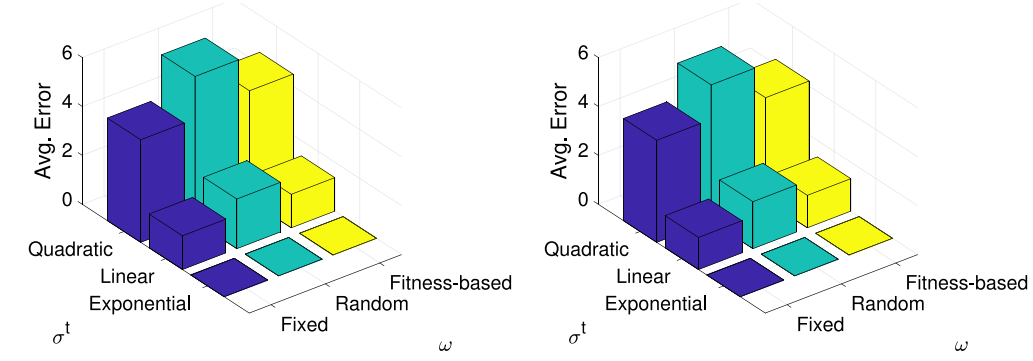
Tables 4 and 5 respectively show the average errors and standard deviations of these competing algorithms in solving 30-dimensional CEC2017 test functions using 30 search agents and in solving 100-dimensional CEC2017 test functions using 100 search agents. In Tables 3 and 4, the first column lists the sequence of 30 benchmark functions in the CEC2017 test suite; the later four columns respectively exhibit the results obtained by four competing algorithms: GWO, FH-GWO, RW-GWO and EGWO. The minimal average error in each contrast group is emphasized in bold.

As shown in Table 4, the GWO algorithm obtained somewhat better results only in solving the 16th and 29th functions; the RW-GWO algorithm acquired slightly better results only in solving functions 7, 17 and 20. Unfortunately, the two algorithms cannot gain a minimal error in those 100-dimensional problems except for the 20th test function, as seen from Table 5. The FH-GWO algorithm never got a better score regardless of solving 30-dimensional or 100-dimensional problems. For the three competitors, so bad performances may be attributed to the search bias of the grey wolf optimizer towards the origin of the coordinate system. We recall that all functions in the CEC2017 test suite are randomly shifted.

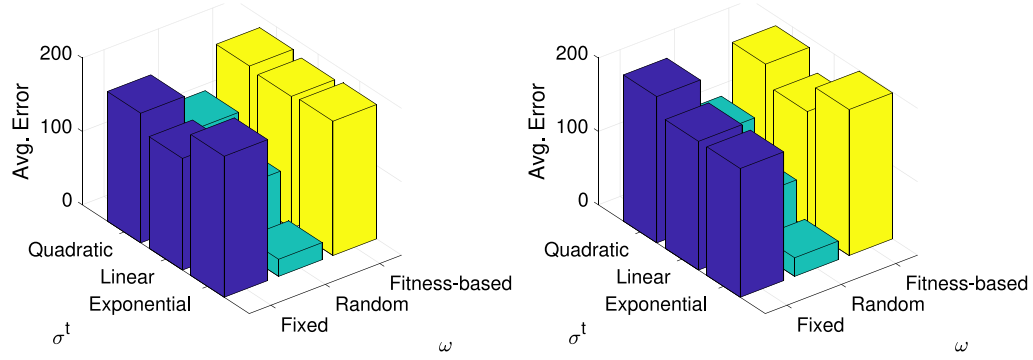
From Tables 4 and 5, we can see that the average errors and standard deviations obtained by the EGWO algorithm are significantly less than those gained by other three competitors in solving the most functions, such as 2, 3, 12, 19 and 30. If we rank these competing algorithms in ascending order of mean error for each test function and then sum these ranks, the summation of the EGWO algorithm is always minimal as shown in the bottom row



(a) f_a: Sphere function (b) f_a^s: Shifted Sphere function



(c) f_b: Schwefel's Problem 1.2 (d) f_b^s: Shifted Schwefel's Problem 1.2



(e) f_c: Rastrigin function (f) f_c^s: Shifted Rastrigin function

Fig. 3. Average errors yielded by the EGWO algorithm in solving the representative problems with 30 dimensions and their shifted variants using various ways to set parameters $\sigma(t)$ and ω over 30 independent runs.

of Tables 4 and 5. That is to say, the performances of the proposed algorithm is absolutely the best in solving real-parameter numerical optimization problems in the CEC2017 suite.

To examine the significance of the differences between the performances of the proposed EGWO algorithm and those of other competing algorithms, the multiple comparison test was carried out. The p-value of the Friedman's test is equal to 4.35E−11 on the 30-dimensional suite and 7.56E−15 on the 100-dimensional suite. The two p-values are both much less than the 5% significance level. This means that there are entire differences between the performances of all competing algorithms. To detect the concrete performance differences among comparative algorithms, the Bonferroni's test was performed. The last row in Tables 4 and 5 shows the post-hoc test results of the proposed EGWO algorithm against algorithms GWO, FH-GWO and RW-GWO. It is seen that all P-values are less than the 5% significance level. The statistical results indicate that the proposed EGWO algorithm significantly outperforms the original GWO algorithm and the two latest variants regardless of solving the 30-dimensional benchmark suite or the 100-dimensional benchmark suite.

The excellent performances of the proposed EGWO algorithm are intuitively exhibited in Fig. 4. This figure plots the average convergence of each competing algorithm on some selected functions over 30 independent runs. The horizontal axis is the number of function evaluation while the vertical axis is the average of errors. In Fig. 4, algorithms GWO, FH-GWO, RW-GWO and EGWO are respectively by a blue dashed line with plus sign markers, a red dotted line with cross markers, an orange dash–dot line with diamond markers and a purple solid line with circle markers. The x-axis records the number of function evaluation and the y-axis is the average error. For readability, these convergence curves exhibit some average errors at points [0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]*NFE, where NFE is the maximal function evaluation number. As a result, each convergence curve has 12 markers and the first point in each curve does not stand for its initial error. As illustrated in this figure, the average convergence speed

Table 4
Mean errors and standard deviations (±Std.) on CEC2017 test functions with dimension $n = 30$.

Fun.	GWO [8]	FH-GWO [31]	RW-GWO [32]	EGWO
1	1.81E+9(1.13E+9)	1.92E+9(1.16E+9)	1.04E+4(5.81E+3)	**5.09E+3**(6.86E+3)
2	7.10E+28(2.82E+29)	3.44E+31(1.88E+32)	1.91E+4(4.34E+4)	**4.35E−3**(6.42E−3)
3	3.11E+4(1.02E+4)	3.71E+4(1.22E+4)	5.26E+0(4.06E+0)	**8.07E−12**(3.41E−11)
4	2.01E+2(8.93E+1)	1.88E+2(4.55E+1)	8.74E+1(2.22E+1)	**5.37E+1**(3.38E+1)
5	9.45E+1(2.37E+1)	1.13E+2(2.31E+1)	9.28E+1(2.59E+1)	**7.26E+1**(1.79E+1)
6	7.45E+0(3.19E+0)	1.28E+1(4.47E+0)	4.61E+0(4.61E+0)	**1.94E−1**(6.00E−1)
7	1.66E+2(4.59E+1)	1.79E+2(3.45E+1)	**8.60E+1**(1.40E+1)	9.91E+1(1.82E+1)
8	8.26E+1(2.25E+1)	9.22E+1(1.73E+1)	8.53E+1(1.70E+1)	**7.01E+1**(1.95E+1)
9	9.83E+2(7.08E+2)	1.19E+3(5.50E+2)	1.91E+2(1.76E+2)	**1.85E+2**(2.69E+2)
10	3.02E+3(9.12E+2)	3.20E+3(5.10E+2)	3.35E+3(5.33E+2)	**2.78E+3**(5.73E+2)
11	8.65E+2(8.75E+2)	9.50E+2(8.13E+2)	1.58E+2(4.52E+1)	**1.19E+2**(5.79E+1)
12	5.88E+7(7.20E+7)	9.93E+7(7.31E+7)	3.83E+6(3.14E+6)	**4.14E+4**(2.22E+4)
13	1.47E+8(7.37E+8)	2.20E+7(8.03E+7)	7.87E+4(4.23E+4)	**1.63E+4**(1.60E+4)
14	1.69E+5(2.91E+5)	3.15E+5(4.88E+5)	1.46E+4(1.43E+4)	**7.80E+3**(5.16E+3)
15	1.39E+6(6.23E+6)	1.23E+6(6.22E+6)	4.78E+4(3.64E+4)	**7.05E+3**(8.11E+3)
16	**7.72E+2**(2.26E+2)	8.12E+2(2.29E+2)	7.73E+2(1.91E+2)	7.89E+2(2.76E+2)
17	3.33E+2(1.57E+2)	2.82E+2(1.21E+2)	**2.65E+2**(1.31E+2)	3.99E+2(2.15E+2)
18	7.76E+5(1.42E+6)	8.87E+5(1.23E+6)	2.05E+5(1.72E+5)	**1.19E+5**(6.49E+4)
19	1.75E+6(5.81E+6)	2.85E+6(6.59E+6)	1.55E+5(1.33E+5)	**9.18E+3**(1.27E+4)
20	3.78E+2(1.27E+2)	4.04E+2(1.65E+2)	**3.62E+2**(1.40E+2)	4.82E+2(2.07E+2)
21	2.84E+2(2.92E+1)	2.93E+2(1.77E+1)	2.86E+2(2.25E+1)	**2.72E+2**(1.81E+1)
22	2.38E+3(1.84E+3)	2.63E+3(1.46E+3)	2.28E+3(1.84E+3)	**1.86E+3**(1.77E+3)
23	4.55E+2(2.32E+1)	4.64E+2(3.86E+1)	4.55E+2(2.59E+1)	**4.12E+2**(2.07E+1)
24	5.45E+2(6.78E+1)	5.22E+2(3.23E+1)	5.32E+2(2.62E+1)	**4.92E+2**(2.14E+1)
25	4.83E+2(4.16E+1)	4.87E+2(3.47E+1)	3.91E+2(1.35E+1)	**3.89E+2**(6.11E+0)
26	2.05E+3(4.59E+2)	2.14E+3(3.36E+2)	2.17E+3(4.68E+2)	**1.68E+3**(4.33E+2)
27	5.51E+2(2.54E+1)	5.53E+2(2.40E+1)	5.41E+2(1.44E+1)	**5.22E+2**(1.06E+1)
28	5.83E+2(4.42E+1)	6.81E+2(2.19E+2)	4.02E+2(1.35E+1)	**3.71E+2**(6.67E+1)
29	**7.59E+2**(1.47E+2)	8.48E+2(1.61E+2)	9.04E+2(1.52E+2)	7.92E+2(2.16E+2)
30	4.73E+6(3.06E+6)	5.10E+6(4.04E+6)	1.45E+6(9.78E+5)	**6.08E+3**(3.16E+3)
S	85	109	66	40
R	3	4	2	1
Pv	4.08E−05	3.12E−11	4.59E−02	

S: The summation of the rank of algorithm in ascending order of mean error for each test function.
R: The rank of the above summation in ascending order.
Pv: The p-value of the Bonferroni's post-hoc test.

of the EGWO algorithm is remarkably faster than that of other competitors. Astonishingly, the proposed algorithm can converge into the optimal solution at the least cost of computation in solving some functions, such as 6 and 14. The superior performances of the proposed algorithm should be attributed to the improved model which takes full advantage of the leader wolves. More precisely, Eq. (5) can provide an effective and efficient estimation for the location of the prey that represents the global optimal solution.

6. Real engineering applications

The proposed algorithm is also applied to two classical engineering optimization problems: tension/compression string design and pressure vessel design, which are often employed as constrained optimization benchmark problems [38–42].

In this paper, the popular penalty function is used to handle constraints in these engineering optimization problems. More precisely,

$$fit(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda \sum_i \max\{0, g_i(\boldsymbol{x})\} \tag{16}$$

where $fit(\boldsymbol{x})$ is the fitness function while $f(\boldsymbol{x})$ is the objective function to be minimized; $g_i(\boldsymbol{x}) \leqslant 0$ is the ith inequality constraint. Notice that any equality should be transformed into a corresponding tolerance-based inequality considering the precise limitation of the computer. In Eq. (16), λ is a penalty factor. The value of the common factor should be a fairly larger number, otherwise the punishment may be inefficient. In this paper, $\lambda = 10^5$.

6.1. Tension/compression string design

The objective of the problem is to minimize the weight of a tension/compression string subject to constraints on minimum deflection, shear stress, surge frequency and limits on outside diameter. This problem refers to three design variables: wire diameter (x_1), mean coil diameter (x_2) and active coil number (x_3). Its mathematical formulation is as follows.

$$\min f(\boldsymbol{x}) = x_1^2 x_2 (x_3 + 2)$$

Subject to

$$g_1(\boldsymbol{x}) = -\frac{x_2^3 x_3}{71785 x_1^4} + 1 \leqslant 0$$

$$g_2(\boldsymbol{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_1^3 x_2 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leqslant 0$$

$$g_3(\boldsymbol{x}) = -\frac{140.45 x_1}{x_2^2 x_3} + 1 \leqslant 0$$

$$g_4(\boldsymbol{x}) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0$$

Variable range:

$$0.05 \leqslant x_1 \leqslant 2$$
$$0.25 \leqslant x_2 \leqslant 1.3$$
$$2 \leqslant x_3 \leqslant 15$$

This problem has been tackled by many other meta-heuristics, such as genetic algorithm [43], particle swarm optimization [38], differential evolution [39], harmony search [40] and grey wolf optimizer [8]. Table 6 lists the best solutions of this problem obtained by these algorithms. As shown in the table, the EGWO algorithm found a optimal solution only using 10 000 iterations approximately consuming 0.089 CPU second. Moreover, the optimal solution found by the proposed EGWO algorithm, $f^*(0.05, 0.374433,$

Table 5
Mean errors and standard deviations (±Std.) on CEC2017 test functions with dimension $n = 100$.

Fun.	GWO [8]	FH-GWO [31]	RW-GWO [32]	EGWO
1	2.77E+10(5.80E+9)	3.48E+10(8.84E+9)	5.49E+5(1.68E+5)	**1.17E+4**(1.66E+4)
2	3.25E+120(1.76E+121)	1.94E+124(1.03E+125)	8.09E+61(4.41E+62)	**1.83E+53**(1.00E+54)
3	2.06E+5(2.29E+4)	2.11E+5(1.68E+4)	3.07E+4(8.26E+3)	**6.53E+3**(7.20E+3)
4	2.53E+3(6.89E+2)	2.62E+3(7.05E+2)	2.77E+2(2.95E+1)	**2.37E+2**(3.25E+1)
5	5.39E+2(5.06E+1)	6.12E+2(4.83E+1)	4.51E+2(6.04E+1)	**3.63E+2**(4.65E+1)
6	2.79E+1(4.84E+0)	3.67E+1(4.49E+0)	2.28E+1(4.78E+0)	**1.40E−1**(1.62E−1)
7	1.00E+3(9.67E+1)	1.22E+3(1.22E+2)	5.49E+2(5.99E+1)	**5.37E+2**(7.69E+1)
8	5.46E+2(6.42E+1)	6.09E+2(6.50E+1)	4.37E+2(5.53E+1)	**3.56E+2**(6.94E+1)
9	2.33E+4(1.08E+4)	1.65E+4(2.68E+3)	8.41E+3(1.92E+3)	**7.95E+3**(4.99E+3)
10	1.33E+4(3.15E+3)	1.55E+4(1.76E+3)	1.36E+4(1.01E+3)	**1.15E+4**(1.23E+3)
11	3.60E+4(1.10E+4)	4.12E+4(9.64E+3)	1.47E+3(2.43E+2)	**1.00E+3**(3.48E+2)
12	5.48E+9(2.60E+9)	5.69E+9(2.24E+9)	1.69E+8(7.92E+7)	**2.05E+6**(1.05E+6)
13	5.75E+8(6.75E+8)	4.25E+8(4.95E+8)	7.10E+4(2.91E+4)	**6.71E+3**(5.08E+3)
14	4.39E+6(3.42E+6)	3.37E+6(2.17E+6)	8.96E+5(3.59E+5)	**1.19E+5**(4.13E+4)
15	5.79E+7(7.49E+7)	9.67E+7(1.45E+8)	5.99E+4(2.38E+4)	**3.89E+3**(4.53E+3)
16	4.02E+3(1.03E+3)	4.26E+3(5.67E+2)	4.30E+3(5.76E+2)	**3.37E+3**(6.23E+2)
17	2.67E+3(5.04E+2)	2.94E+3(4.71E+2)	3.26E+3(4.43E+2)	**2.53E+3**(4.79E+2)
18	4.04E+6(2.16E+6)	3.62E+6(2.40E+6)	1.27E+6(6.24E+5)	**8.40E+5**(4.05E+5)
19	5.54E+7(4.75E+7)	8.36E+7(1.22E+8)	3.92E+6(1.90E+6)	**4.43E+3**(5.23E+3)
20	**2.36E+3**(6.86E+2)	2.65E+3(4.37E+2)	3.01E+3(4.28E+2)	2.52E+3(4.63E+2)
21	7.58E+2(5.69E+1)	8.30E+2(6.83E+1)	6.89E+2(6.39E+1)	**6.06E+2**(4.22E+1)
22	1.51E+4(1.34E+3)	1.70E+4(1.55E+3)	1.47E+4(1.14E+3)	**1.26E+4**(1.25E+3)
23	1.08E+3(7.34E+1)	1.15E+3(6.70E+1)	1.09E+3(6.55E+1)	**8.11E+2**(4.31E+1)
24	1.51E+3(8.68E+1)	1.63E+3(1.08E+2)	1.53E+3(1.07E+2)	**1.23E+3**(7.56E+1)
25	2.83E+3(5.43E+2)	3.01E+3(4.87E+2)	8.23E+2(5.48E+1)	**7.50E+2**(7.09E+1)
26	1.00E+4(8.98E+2)	1.08E+4(1.02E+3)	9.82E+3(9.49E+2)	**6.99E+3**(4.42E+2)
27	1.16E+3(1.26E+2)	1.18E+3(9.78E+1)	1.05E+3(8.81E+1)	**7.50E+2**(4.52E+1)
28	3.56E+3(9.71E+2)	3.90E+3(9.64E+2)	6.36E+2(4.32E+1)	**5.82E+2**(3.27E+1)
29	4.36E+3(6.52E+2)	4.63E+3(4.89E+2)	5.12E+3(4.92E+2)	**3.08E+3**(4.95E+2)
30	3.57E+8(3.52E+8)	3.63E+8(2.28E+8)	3.86E+7(1.53E+7)	**7.15E+3**(3.42E+3)
S	86	112	71	31
R	3	4	2	1
Pv	2.28E−07	3.30E−15	3.80E−04	

Notations S, R and Pv are the same as those in Table 4.

Table 6
Comparison of the best solutions found by EGWO and other algorithms in solving the tension/compression spring design problem.

Algorithm	f^*	x_1	x_2	x_3	nfe
GA [43]	0.012681	0.051989	0.363965	10.89052	80 000
IAPSO [38]	0.012665	0.051685	0.356629	11.29417	20 000
CDE [39]	0.012670	0.051609	0.354714	11.41083	204 800
IPHS [40]	0.012666	0.051861	0.360858	11.05034	200 000
GWO [8]	0.012666	0.051690	0.356737	11.2885	N/A
RW-GWO [32]	0.012674	0.05167	0.35613	11.33056	30 000
EGWO	**0.009872**	**0.05**	**0.374433**	**8.546571**	10 000

nfe: the number of function evaluation.

$8.546571) = 0.009872$, is better than the best known solution so far.

6.2. Pressure vessel design

The objective of the problem is to minimize the total production cost of a cylindrical vessel by four design variables: shell thickness (x_1), spherical head thickness (x_2), inner radius (x_3) and shell length (x_4). Its mathematical formulation can be stated as follows.

$$\min f(\boldsymbol{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

Subject to

$$g_1(\boldsymbol{x}) = -x_1 + 0.0193x_3 \leqslant 0$$
$$g_2(\boldsymbol{x}) = -x_2 + 0.00954x_3 \leqslant 0$$
$$g_3(\boldsymbol{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leqslant 0$$

Table 7
Comparison of the best solutions found by EGO and other algorithms in solving the pressure vessel design problem with the original variable range.

Algorithm	f^*	x_1	x_2	x_3	x_4	nfe
GA [42]	7207.494	1.125	0.625	58.1978	44.293	N/A
IPHS [40]	7197.73	1.125	0.625	58.29014	43.69276	100 000
EGWO	**7021.126**	**1.101507**	**0.600**	**57.07285**	**50.55135**	10 000

nfe: the number of function evaluation.

Variable range:

$$1.1 \leqslant x_1 \leqslant 10$$
$$0.6 \leqslant x_2 \leqslant 10$$
$$40 \leqslant x_3 \leqslant 80$$
$$20 \leqslant x_4 \leqslant 60$$

The EGWO algorithm solved the original version of the problem depicted in [41]. As seen from Table 7, the proposed algorithm obtained a better solution at a lower computation cost of 0.8713 CPU second than that reported in [42] and [40].

Coello [43], Huang [39], Guedria [38], Mirjalili [8] and Gupta [32] et al. also solved the problem using the different search range: $0 \leqslant x_1, x_2 \leqslant 99$ and $10 \leqslant x_3, x_4 \leqslant 200$. Table 8 reports their optimal solutions. As seen from table, the proposed algorithm still achieved the best solutions. Furthermore, the optimal solution obtained by the proposed algorithm, $f(0.7926900, 0.3918270, 41.072022, 189.78466) = 5910.6207$, is better than the best known solution so far.

7. Conclusion

This paper presents an enhanced grey wolf optimizer. Compared with the original version, the new optimizer (1) highlights

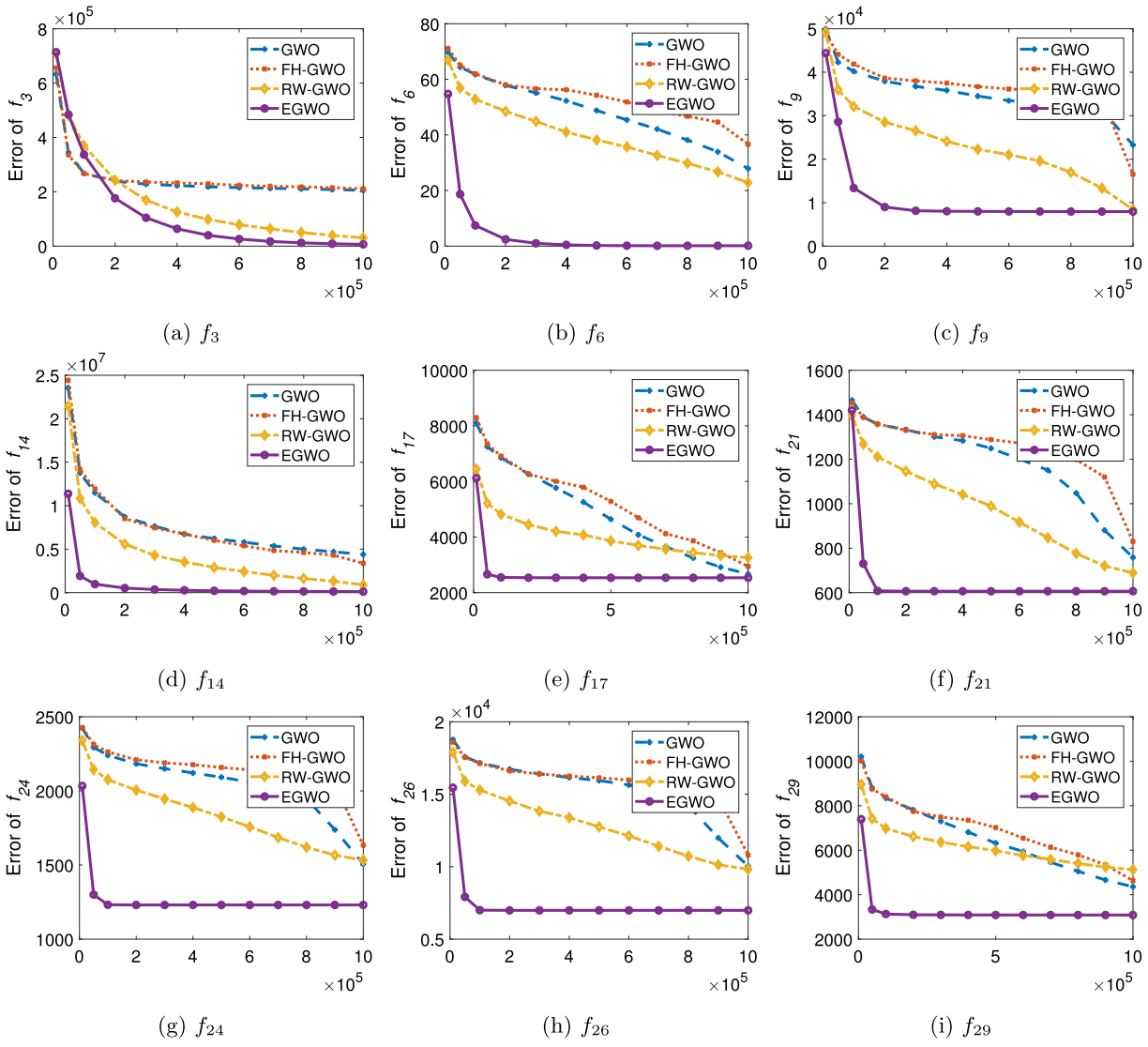


Fig. 4. Average convergence graphs of competing algorithms: GWO, FH-GWO, RW-GWO and EGWO on some selected 100-dimensional functions over 30 independent runs.

Table 8
Comparison of the best solutions found by EGO and other algorithms in solving the pressure vessel design problem with the widening variable range.

Algorithm	f^*	x_1	x_2	x_3	x_4	nfe
GA [43]	6059.9463	0.812500	0.437500	42.097398	176.654050	N/A
DE [39]	6059.7340	0.812500	0.437500	42.089411	176.637690	N/A
IAPSO [38]	6059.71433	0.812500	0.437500	42.0984	176.6366	20 000
GWO [8]	6051.5639	0.812500	0.434500	42.089181	176.758731	N/A
RW-GWO [32]	6059.736	0.81250	0.43750	42.09840	176.63784	300 000
EGWO	**5910.6207**	**0.7926900**	**0.3918270**	**41.072022**	**189.78466**	10 000

nfe: the number of function evaluation.

the main feature of the social leadership hierarchical consideration through the distinct weight; (2) dynamically estimates the potential location of the prey through an information fusion for leader wolves; (3) guides each wolf moving directly towards the estimated location of the prey; (4) properly mimics the random walk behavior of the grey wolf pack except for approaching prey, encircling and attacking. In addition, the experimental result shows that the new optimizer does not again have the significant search bias towards the origin of the coordinate system. Moreover, the comparative results on the CEC2017 test suite indicate that the new optimizer significantly outperforms the original version and two latest variants in terms of the convergence speed and the quality of solution found. The proposed optimizer is also successfully applied to two engineering optimization benchmark problems: the tension/compression string design problem and the pressure vessel design problem. It is worth emphasizing that the enhanced grey wolf optimizer breaks the records of the best known solutions of the two real-world optimization problems at a lower computation cost.

This paper reports a new discovery that the original grey wolf optimizer has a strong search bias towards the origin of the coordinate system. This shortage might exist in the binary and multi-objective versions and even other meta-heuristics. Thus, the

improvement for the search bias would be the future research direction.

In addition, this paper investigates the control parameter selection for the new optimizer and gives a suggestion in form. However, there is still a room to improve by adaptive or self-adaptive parameter fine-tuning.

Acknowledgments

The author would like to thank the associate editor and anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China under grant nos. 91646110, 71871004 and 71571021.

References

[1] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization: an overview, Swarm Intell. 1 (1) (2007) 33–57.
[2] M. Dorigo, C. Blum, Ant colony optimization theory: a survey, Theoret. Comput. Sci. 344 (2) (2005) 243–278.
[3] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[4] I. Fister, X.-S. Yang, J. Brest, A comprehensive review of firefly algorithms, Swarm Evol. Comput. 13 (2013) 34–46.
[5] X.-S. Yang, D. Suash, Cuckoo search: recent advances and applications, Neural Comput. Appl. 24 (1) (2014) 169–174.
[6] S. Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98.
[7] J.J.Q. Yu, V.O.K. Li, A social spider algorithm for global optimization, Appl. Soft Comput. 30 (2015) 614–627.
[8] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[9] E. Gupta, A. Saxena, Robust generation control strategy based on grey wolf optimizer, J. Electr. Syst. 11 (2) (2015) 174–188.
[10] A. Chaman-Motlagh, Superdefect photonic crystal filter optimization using grey wolf optimizer, IEEE Photonics Technol. Lett. 27 (22) (2015) 2355–2358.
[11] M.H. Sulaiman, Z. Mustaffa, M.R. Mohamed, O. Aliman, Using the gray wolf optimizer for solving optimal reactive power dispatch problem, Appl. Soft Comput. 32 (2015) 286–292.
[12] A.A. El-Fergany, H.M. Hasanien, Single and multi-objective optimal power flow using grey wolf optimizer and differential evolution algorithms, Electric Power Compon. Systems 43 (13) (2015) 1548–1559.
[13] X. Song, L. Tang, S. Zhao, X. Zhang, L. Li, J. Huang, W. Cai, Grey wolf optimizer for parameter estimation in surface waves, Soil Dyn. Earthq. Eng. 75 (2015) 147–157.
[14] G.M. Komaki, V. Kayvanfar, Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time, J. Comput. Sci. 8 (2015) 109–120.
[15] S. Mirjalili, How effective is the grey wolf optimizer in training multi-layer perceptrons, Appl. Intell. 43 (1) (2015) 150–161.
[16] E. Emary, H.M. Zawbaa, A.E. Hassanien, Binary grey wolf optimization approaches for feature selection, Neurocomputing 172 (2016) 371–381.
[17] S.A. Medjahed, T.A. Saadi, A. Benyettou, M. Ouali, Gray wolf optimizer for hyperspectral band selection, Appl. Soft Comput. 40 (2016) 178–186.
[18] C. Lu, S. Xiao, X. Li, L. Gao, An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production, Adv. Eng. Softw. 99 (2016) 161–176.
[19] S. Zhang, Y. Zhou, Z. Li, W. Pan, Grey wolf optimizer for unmanned combat aerial vehicle path planning, Adv. Eng. Softw. 99 (2016) 121–136.
[20] T. Jayabarathi, T. Raghunathan, B.R. Adarsh, P.N. Suganthan, Economic dispatch using hybrid grey wolf optimizer, Energy 111 (2016) 630–641.
[21] M.A. Tawhid, A.F. Ali, A hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function, Memetic Comput. 9 (4) (2017) 347–359.
[22] A. Kumar, S. Pant, M. Ram, System reliability optimization using gray wolf optimizer algorithm, Qual. Reliab. Eng. Int. 33 (7) (2017) 1327–1335.
[23] R.-E. Precup, R.-C. David, E.M. Petriu, Grey wolf optimizer algorithm-based tuning of fuzzy control systems with reduced parametric sensitivity, IEEE Trans. Ind. Electron. 64 (1) (2017) 527–534.
[24] A.K.M. Khairuzzaman, S. Chaudhury, Multilevel thresholding using grey wolf optimizer for image segmentation, Expert Syst. Appl. 86 (2017) 64–76.
[25] S. Mirjalili, S. Saremi, S.M. Mirjalili, L.D.S. Coelho, Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization, Expert Syst. Appl. 47 (2016) 106–119.
[26] A. Sahoo, S. Chandra, Multi-objective grey wolf optimizer for improved cervix lesion classification, Appl. Soft Comput. 52 (2017) 64–80.
[27] W. Long, J. Jiao, X. Liang, M. Tang, An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization, Eng. Appl. Artif. Intell. 68 (2018) 63–80.
[28] S. Saremi, S.Z. Mirjalili, S.M. Mirjalili, Evolutionary population dynamics and grey wolf optimizer, Neural Comput. Appl. 26 (5) (2015) 1257–1263.
[29] N. Mittal, U. Singh, B.S. Sohi, Modified grey wolf optimizer for global engineering optimization, Appl. Comput. Intell. Soft Comput. (8) (2016) 1–16.
[30] A.A. Heidari, P. Pahlavani, An efficient modified grey wolf optimizer with lévy flight for optimization tasks, Appl. Soft Comput. 60 (2017) 115–134.
[31] L. Rodr I Guez, O. Castillo, J.E. Soria, P. Melin, F. Valdez, C.I. Gonzalez, G.E. Martinez, J. Soto, A fuzzy hierarchical operator in the grey wolf optimizer algorithm, Appl. Soft Comput. 57 (2017) 315–328.
[32] S. Gupta, K. Deep, A novel random walk grey wolf optimizer, Swarm Evol. Comput. 44 (2019) 101–112.
[33] M. Davarynejad, J. van den Berg, J. Rezaei, Evaluating center-seeking and initialization bias: the case of particle swarm and gravitational search algorithms, Inform. Sci. 278 (2014) 802–821.
[34] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248.
[35] S. Garc I A, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the CEC2005 special session on real parameter optimization, J. Heuristics 15 (6) (2009) 617–644.
[36] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[37] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Tech. rep., Nanyang Technological University, 2016, pp. 1–34.
[38] N.B. Guedria, Improved accelerated PSO algorithm for mechanical engineering optimization problems, Appl. Soft Comput. 40 (2016) 455–467.
[39] F.-z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (1) (2007) 340–356.
[40] M. Jaberipour, E. Khorram, Two improved harmony search algorithms for solving engineering optimization problems, Commun. Nonlinear Sci. Numer. Simul. 15 (11) (2010) 3316–3331.
[41] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (2) (1990) 223–229.
[42] S.-J. Wu, P.-T. Chow, Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization, Eng. Optim. 24 (2) (1995) 137–159.
[43] C.A. Coello Coello, E.E.N. Mezura Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inform. 16 (3) (2002) 193–203.