Grey Wolf optimization algorithm with random local optimal regulation and first-element dominance

Abstract

Due to the classical Grey Wolf algorithm GWO does not consider the characteristics of the local information of individual in population, a novel local random optimization strategy is proposed to make up for the defect of GWO. In this method, several points in the neighborhood of the current location of each individual are selected at random in the axial direction as candidates, and the best points are selected to participate in the renewal decision of the individual. Furthermore, in our experiments, a special first-element dominance characteristic is found and can greatly improve the combination effect of global and local information. In order to ensure that all constraints are not violated in the process of constraint optimization in industrial design, the random mixed population initialization method is proposed to generate population individuals that meet the constraint requirements and contain boundary values randomly. In addition, a treatment method of shrinking in a specific direction is proposed for dealing with individuals who cross the boundary. Experimental results on several test function sets show that compared with recent improved algorithms for GWO, the proposed algorithm has obvious advantages in fitness value, convergence speed and stability .

Keywords: intelligence algorithm, grey Wolf algorithm, first-element dominance, Random Local Optimization, Random mixed population.

1. Introduction

A large number of optimization problems have emerged in scientific research and engineering applications, and these problems often have high dimensionality. If the problem to be solved is non-convex, the traditional optimization algorithm is often prone to falling into the local optimal solution, and it is difficult to obtain the global optimal solution. Bio-inspired optimization algorithm provides a feasible path to solve practical optimization problem. It is widely used in mechanical engineering [1,2], electronic engineering [3], computer vision [4,5], architectural design [6], biological information [7], a-griculture [8], financial markets [9] and so on. Due to the wide range of applications of Bio-inspired optimization algorithm, a large number of algorithms have gushed out, including: (1) Evolutionary algorithm, inspired from Darwinian Theory of Evolution, simulate evolution rules in nature, of which genetic algorithm (GA) [10] and differential evolution (DE) [11] are typical representatives of such algorithms. (2) Swarm intelligence optimization method, which is mainly inspired by the social behavior of the populations in the fauna, and shares the information of all individuals in the optimization process. They include: Particle Swarm Optimization (PSO) [12], Firefly Algorithm (FA) [13], Grey Wolf Optimizer (GWO) [14], Ant Colony Optimization Algorithm (ACO) [15], Artificial Bee Colony Algorithm (ABC) [16,17], Whale Optimization Algorithm (WOA) [18], and Marine Predator Algorithm (M-PA) [19], Cuckoo Search Algorithm[20], Honey Badger Algorithm[21], Orca Predation Algorithm[22], Gazelle Optimization Algorithm[23], Salp Search Algorithm[24] and so on. (3) Based on human behavior optimization methods, inspired by human interaction phenomena or a human society behavior. Examples include the Teaching and Learning Optimization Algorithm (TLBO) [25], the Empire Competition Algorithm (ICA) [26], as well as the Volleyball Super League Algorithm (VPL) [27] and the Cultural Evolution Algorithm (CEA) [28],slime mould algorithm(SMA)[29].

Among many bio-inspired optimization algorithms, swarm intelligent optimization algorithm is the most favored by researchers and occupies the highest proportion in optimization algorithms. Swarm intelligence algorithms pay attention to the sociality of groups and the information sharing among individuals. The differences of various swarm intelligence algorithms are mainly reflected in the different ways of information sharing, and each has its own scope of application. Among them, Grey Wolf Optimization Algorithm (GWO) has been reported for its simplicity and efficiency. Sen Zhang [30] integrated Powell local optimization method into GWO and proposed PGWO algorithm. The algorithm first used GWO to generate a new position of each individual and then used Powell local optimization method to adjust the position of each individual. Assen Beshr Alyu [31] embedded particle swarm optimization (PSO) into GWO, proposed hybrid Grey Wolf optimizer and particle swarm optimization algorithm (PSO-GWO), and applied them to determine the optimal location and scale of distributed generation (DG) in passive distribution systems, while considering multi-objective functions. Tt includes minimizing of active and reactive power losses, and enhaning voltage distribution. In this method, the three individual revisions obtained from GWO are used to correct the individual positions in PSO. Chao Lu et al. [32] used chaos to optimize two important decay control parameters in GWO. M.H. Nadimi-Shahraki et al. [33] proposed the I-GWO algorithm. Based on classical GWO, the Euclidean distance between the current position and the new position calculated by GWO algorithm was used as the radius to construct the individual neighborhood. The classical GWO results were modified by using the difference between the randomly selected populations within and outside the neighborhood. Yilin Su et al. [34] used various strategies to optimize GWO. Firstly, the initial population of GWO was initialized by using the combination model of logistic and Lotka-Volterra to improve the difference of the initial population. Then, the population renewal factor was adjusted by nonlinear function to improve the search efficiency. Kewen Li et al. [35] designed the Cauchy-Gauss mutation operator, which was applied to the alpha wolf. When the leading wolf tends to the local optimal solution, the search range can be expanded, and the operator can effectively improve the local development ability of the leading wolf and avoid falling into the local optimal solution. A greedy selection mechanism is proposed, to avoid high population diversity caused by variation. Greedy selection mechanism can maintain the diversity of the population and ensure the convergence speed of the algorithm. An improved search strategy is designed for all grey wolf individuals, which takes into account the average position of all individuals, effectively expands the search space, and improves the global search ability of the algorithm. Ashish Kumar Tripathi et al. [36] proposed the Enhanced Grey Wolf Optimizer (EGWO), which hybridized the hunting strategy of grey wolves with binomial crossing and introduced arbitrary flying steps to enhance the hunting ability of grey wolves. Nikhil Paliwal et al. [37] applied evolutionary intelligence-based the Grey Wolf Optimizer (GWO) to estimate the optimal parameters of proportional-integral-derivative (PID), and multi-source single-area Load Frequency Control (LFC) controller. The multi-source single-area power network considered in this paper consists of reheat turbine thermal power plant, gas power plant and hydraulic power plant with mechanical hydraulic governor. Furthermore, GWO is used to optimize the PID controller parameters of the LFC system under study. The algorithm takes into account four important performance indicators, namely, time-weighted square error integral (ITSE), time-weighted absolute error integral (ITAE), square-error integral (ISE) and absolute error integral (IAE). Finally, the results of the proposed method are compared with those of genetic algorithm (GA) under 1%, 2% and 5% load perturbations. And the effectiveness and applicability of the proposed method in multi-source single-area network LFC are verified. In order to improve the performance of the basic Grey Wolf Optimization algorithm, Yinqiu Song et al. [38] introduced contraction, elastic surrounding and weighted candidate mechanisms to make up for the defects of local stagnation and premature convergence of the proposed elastic Grey Wolf Optimization algorithm. Jingkai Cui et al. [39] proposed a team-based Grey Wolf Optimizer (TL-GWO), which consists of two strategies. The neighbor learning strategy introduces the influence of neighbors and improves the local search ability, while the random learning strategy provides a new search direction and enhances the global search ability. The Sine-Cosine Algorithm (SCA) proposed by Shubham Gupta et al. [40] is a new member in the field of meta-heuristics, which uses the behavior of sine and cosine functions to find solutions to optimization problems. However,In some cases SCA skips the real solution and falls into a suboptimal one. The problems lead to premature convergence, which is not conducive to the determination of global optimal solutions. Therefore, in order to alleviate the above problems, it establishes a relatively good cooperative relationship between regional development and development. Firstly, the exploration capability of SCA is improved by integrating social and cognitive components; Secondly, the balance between exploration and exploitation is maintained through the Grey Wolf Optimizer (GWO). This algorithm is named SC-GWO. Chengsheng Pan et al. [41] reanalyzed the hunting behavior of wolves and observed that the communication between leaders and ω wolves was very frequent during the hunt. This process is called judging prey. Based on this, an improved optimization algorithm called Decision Grey Wolf Optimization algorithm (DGWO) is proposed. Unlike the original GWO, DGWO consists of four steps instead of three: finding the prey, judging the prey, surrounding the prey, and attacking the prey. Bingkun Wang et al. [42] proposed a variant of GWO, called the Cross-Dimensional Coordinated Grey Wolf Optimizer (CDCGWO). It utilizes a novel learning technique to update the best solution (prey location) using all the previous best knowledge obtained from the frank solution. This method preserves the diversity of wolves and prevents premature convergence of multimodal optimization tasks. In addition, CDCGWO provides a unique constraint management method for realistic constrained engineering optimization problems. CDCGWO is used for evaluation in multiple engineering domains (industrial chemical production, power systems, process design and synthesis, mechanical design, power electronics and livestock feed distribution) and performance on multiple test problems (15 widely used multimodal numerical test functions, 10 complex IEEE CEC06-2019 clothing tests, randomly generated landscapes, and 12 constrained reality optimization problems). The ENGWO algorithm proposed by Gyanaranjan Shial et al. [43] introduces a differential perturbation operator into the grey Wolf optimization algorithm,and randomly selects three Omega wolves to help the three leading wolves of the original algorithm achieve diversification of solution quality among feasible Omega wolves. In addition, when updating the position of individual omega wolves, similar values are used for the control parameters (A and C) of each leader Wolf's GWO. To ensure this diversity among omega wolves, an exploration element is introduced during the development phase, further improving the optimization capabilities of the GWO algorithm. The LMWOAGWO algorithm proposed by Qian Yang et al. [44] integrates WOA algorithm into GWO, adopts pseudo-inverse strategy to randomly initialize the population, introduces chaotic strategy to optimize the individual position update mechanism, and adopts Lvy flight technology to improve the global search capability of the algorithm.Recently,Kai Zhouet et al.[45] proposed an adaptive gray wolf fast optimization algorithm (SS-GWO), which adaptively selects the global search or local search according to the degree of agglomeration of individuals. Oluwatayomi Rereloluwa Adegboye et al.[46] proposed a algorithm called CMWGWO,it incorporates innovative approaches such as Chaotic Opposition Learning (COL), Mirror Reflection Strategy (MRS), and Worst Individual Disturbance (WID) into GWO.

According to the current research on GWO, the main focus is using local information to fine-tune GWO algorithm results or adjust the interactions between individuals in a population. From the reported literature, it can be seen that these improvements can optimize the performance of the original GWO algorithm. It is known that the determining factors for each individual moving from the current position to the next position are the direction of movement and the distance of movement. At present, most swarm intelligence algorithms adopt the strategy of gradually decreasing the moving distance with the number of iterative steps. That is, the moving distance is a decreasing function of the number of iterative steps. It is generally set before the population evolution and is not affected by the current situation of the population. Even if there is interference, the moving distance in the current iteration step is randomly fine-tuned based on the current population state. However, the direction of movement is constantly adjusted as the population evolves, sometimes by a large margin. In classical GWO, the movement direction of each individual is mainly determined by the position of the three individuals with the highest score, which reflects the global relationship of individuals in the population, while ignoring the local characteristics of individuals. Inspired by Powell local optimization, this paper proposes a new idea that global optimal and local optimal co-determine the movement direction of each individual, and proposes new strategies for population initialization and transboundary processing for real-world constrained optimization problems.

The specific contributions of this paper and the differences of our work with these work are as follows:

(1) Difference of the local searching strategy: the proposed algorithm randomly selects a small number of dimensions from all the coordinate dimensions of the current individual, and randomly selects a point in an interval distance r (gradually decrease with iterative process.) from the current individual location on the selected coordinate dimension as an alternative point, and selects the best point from all the alternative points as the local update value of the current individual. Its advantages are: the randomness of the search, the search range is gradually reduced with the number of iteration steps, and the calculation cost is reduced.

(2) Difference of Individual renewal mechanism: The proposed algorithm combines the classical GWO calculation results with the proposed random local search results, and selects the best of the 6 combinations as the final update value of the current individual.

(3) A combination mechanism called first-element dominance is found, that is, in the partially weighted combination in (2), the classical GWO calculation results participate in the combination with its first-element form. Experiments show that this method can greatly improve the performance of the algorithm.

(4) To solve the constrained optimization problem, a random hybrid population generation method is proposed to ensure that the initial population satisfies the constraints and boundary conditions, and at the same time, some dimensions of some individuals are randomly selected at the boundary of the search domain.

(5) To solve the constraint optimization problem, a new individual updating strategy is proposed to ensure that the individual can continue to meet the constraint and boundary requirements when the position is updated.

2. Relevant research basis

In order to facilitate the subsequent expression and comparison, this section introduces the classical Grey Wolf algorithm.

In the GWO algorithm, α, β and δ perform the pursuit behavior, and ω follows them to track and suppress prey, and finally complete the predation task. When GWO algorithm is used to solve the continuous function optimization problem, it is assumed that the number of grey wolves in the grey wolf population is N and the search space is d dimension. The position of the i-th grey wolf in d dimensional space can be expressed as $x^{(i)} = (x_1^{(i)}, \cdots, x_d^{(i)})$. The current optimal individual in the population is denoted as α, the corresponding individuals with the second and third fitness values are denoted as β and δ, the rest of individuals are denoted as ω, and the prey position corresponds to the global optimal solution of the optimization problem.

During predation, the grey Wolf first needs to surround the prey. In GWO algorithm, the corresponding distance between the individual and the prey needs to be determined, here the corresponding distance is calculated according to equation (1).

$$D = |C \times X_p(t) - X(t)| \tag{1}$$

Where, $X_p\,(t)$ represents the position of prey $p\,(p \in \{\alpha, \beta, \delta\})$ in the t generation, and $X(t)$ represents the position of individual grey wolf in the t generation, the constant C is the wobble factor and is determined by the following equation (2).

$$C = 2r_1 \tag{2}$$

Where r_1 is a random number in $[0, 1]$.

grey wolf location updated by equation (3).

$$X_p\,(t+1) = X_p\,(t) - A \times D \tag{3}$$

Where A is the convergence factor, which is determined by the following equation (4).

$$A = 2ar_2 - a \tag{4}$$

Where r_2 is a random number in $[0, 1]$, and a is linearly decreasing from 2 to 0 as the number of iterations increases.

When the grey wolf determines the location of the prey, the α wolf will lead the β and δ, and launch the hunt for prey. In wolves, α, β and δ are closest to the prey, and their positions can be used to determine the location of the prey. It can be calculated according to equations 5-11.

$$D_\alpha = |C_1 \times X_\alpha(t) - X(t)| \tag{5}$$

$$D_\beta = |C_2 \times X_\beta(t) - X(t)| \tag{6}$$

$$D_\delta = |C_3 \times X_\delta(t) - X(t)| \tag{7}$$

$$X_1 = X_\alpha - A_1 \times D_\alpha \tag{8}$$

$$X_2 = X_\beta - A_2 \times D_\beta \tag{9}$$

$$X_3 = X_\delta - A_3 \times D_\delta \tag{10}$$

$$G\_m\,(t+1) = \frac{1}{3}(X_1 + X_2 + X_3) \tag{11}$$

Where $C_1,\ C_2, C_3, A_1, A_2, A_3$ are generated by equations 2,4.

3. The grey wolf algorithm optimized by Local optimum

In the original GWO algorithm, the movement direction and movement distance of each individual in the population were determined by the location information of the three grey wolves with the highest fitness value in the population. Therefore, the three individuals with the highest scores at each moment determine the evolution direction of the whole population. If these three individuals fall into the local optimal, the probability of the whole population falling into the local optimal increases dramatically, which can also be verified by the experimental results. The optimization results of unimodal functions are obviously much better than those of multi-modal functions. Although the control parameters A and C can help the algorithm escape from local optimality, this influence gradually decreases with the increase of iterations. As a result, the algorithm is more sensitive to the initial population state when dealing with some multi-modal functions, and it is easier to fall into local optimality.

The Powell local search method[30] can be regarded as an extension of the classical Newton iteration method, which changes the traditional single-direction gradient search to multi-direction search, improves the performance of gradient descent method, and is prone to falling into local optimality. However, its local characteristics are obvious, and the search is relatively time-consuming, so it is more likely to fall into local optimality when solving multi-peak functions. If we compare the optimization process with the human growth process, GWO focuses on individual sociality and ignoring individual evolution, while Powell local search only focuses on individual evolution without considering individual sociality. The two are obviously complementary. Therefore, how to make use of their complementarity and reduce the time cost caused by Powell's iterative search is a topic worth studying. However, the practice of using Powell's local search only on the basis of individual location obtained by GWO may greatly change the guiding effect of the optimal individual in GWO on the movement direction of other individuals, or even in the opposite direction, thus reducing the sociality of individuals. Although the iterative process of Powell's local search is time-consuming and highly dependent on the initial direction, its multi-direction local search idea can be used to improve the individual's own evolutionary ability in GWO, so as to make up for the fact that GWO only focuses on sociality and lacks the individual's own learning characteristics.

3.1. Random local search

In order to make full use of local information and complete local optimal search with minimal cost, the proposed random local search, retains the search strategy along dimensions in Powell, but abandons its optimization process of iterative search along dimensions. For each individual in the population, its renewal candidate is located on a randomly selected dimension, and the distance length to that individual is r, Here r decreases as the number of iteration steps increases, and the position is randomly perturbed. Then choose the best one from D candidates as the next update position of the individual. Finally, in each iteration of GWO, the individual update value in the population is jointly determined by the random local search update value and the global update value obtained by GWO.

The number of random search directions is set as D, the search radius is denoted as r, which decreases with the number of iterative steps of the algorithm, the current individual position is denoted as x, its dimension is n, and the fitness function is $fobj()$.The proposed random local search method can be expressed as algorithm 1:

Algorithm 1 Random local search method

Input: individual position x, searching radius r, The number of random search directions D.

Output: Local update value $x^{(j)}$.

Step 1: Randomly generate D search directions (that is, randomly select D dimension from n coordinate dimensions), denoted as $d_1, \cdots, d_D$.

Step 2: Compute:

$$x^{(i)} = x \pm rand() \times r \times d_i, \quad i = 1, 2, \cdots, D. \tag{12}$$

$$j = \arg\min_i fobj(x^i). \tag{13}$$

Step 3: Return $x^{(j)}$.

In each iteration of GWO, the random local search method is used to calculate the local optimal value of each individual x, and the local update value of x is calculated according to Equation (14) :

$$L\_m = |Cx^{(j)} - x| \tag{14}$$

Where $C = 2.0rand(1, dim)$.

3.2. GWO algorithm with Random local search and first-element dominance

In view of the defect that classical GWO algorithm does not consider individual local information, which may lead to local optimization easily, this paper tries to integrate the local search results given in Section 3.1 into the classical GWO search results, so as to reduce the possibility of GWO search falling into local optimization.In the proposed algorithm, we combine the global optimal value obtained from classical GWO and the local optimal value obtained from local search by a simple weighted combination method.

Assuming that the global update value obtained by GWO denotes as G_m, L_m and G_m can be combined according to the formula $\lambda_1 G\_m + \lambda_2 L\_m$ to obtain different update values of individual x. However, in our experiment, it is difficult to find uniform weights λ_1 and λ_2 to make their corresponding weighted combinations adapt to different problems. In order to facilitate calculation and not increase the calculation cost too much, on the basis of many experiments, this paper selects 6 groups of weights, which are respectively (1,0),(1,1),(0.5,0.5),(0.6,0.4),(0.7,0.3),(0.8,0.2). In each iteration, the optimal combination selected from the 6 weighted combinations serves as the new position value of the current individual.

In our algorithm, the 6 combinations listed in Equation (15) are used to calculate the candidate update value of x.

$$\begin{cases} tx^{(1)} = G\_m; \\ tx^{(2)} = G\_m(1) + L\_m; \\ tx^{(3)} = 0.5G\_m(1) + 0.5L\_m; \\ tx^{(4)} = 0.6G\_m(1) + 0.4L\_m; \\ tx^{(5)} = 0.7G\_m(1) + 0.3L\_m; \\ tx^{(6)} = 0.8G\_m(1) + 0.2L\_m. \end{cases} \tag{15}$$

Then the fitness value of each combination is calculated, and the combination with the smallest value is the final update value of the individual x. Namely,

$$j = \arg\min_{1\leq i\leq 6} \{fobj(tx^{(i)})\} \tag{16}$$

$tx^{(j)}$ is the updated value of the current individual x.

It can be seen from Equation (15) that when combining with the local update value, we do not use $\lambda_1 G\_m + \lambda_2 L\_m$, but $\lambda_1 G\_m(1) + \lambda_2 L\_m$. Here $G\_m(1)$ denotes the first element of G_m, The first element of the update value obtained by classical GWO is combined with the local update value to produce a set of vectors, and the vector with the best fitness is selected as the final position update value of the population individual. Why does substituting $G\_m(1)$ for G_m in a combination yield better results? We haven't found a theoretical basis for this yet, it's just that we found this phenomenon in experiments.The experimental results given in Table 8 in Section 4.5 show that the first-element combination has obvious advantages, especially in the first 15 functions among the 23 basic test functions. The propose GWO algorithm with Random local search and first-element dominance can be described as algorithm 2.

Algorithm 2 GWO algorithm with Random local search and first-element dominance

Input: population size N,variable number dim,lower bound lb,upper bound ub,maximum iteration number $MaxIter$,initial local search radius r,random local search direction number dn.

Output: best individual position, best individual fitness.

Step 1:initializing population x

Step 2:computing the fitness of each individuals in population Fit.

Step 3:Three individuals with the best fitness were selected from the population and recorded as α,β and δ respectively.

```
Step 4: set $t = 1$.
Step 5:while $t < MaxIter$
  Step 5.1: set $r = r/t^2$,$a = 2 - t * ((2)/(MaxIter))$.
  Step 5.2:for each individual $x_i$ in population
    Step 5.2.1:computing the local updated value $L\_m$ of $x_i$ by Equation (14).
    Step 5.2.2:computing the global update value $G\_m$ of $x_i$ by Equation (1-11).
    Step 5.2.3:computing the global update value $tx^{(1)}, \cdots, tx^{(6)}$ and $j$ by Equation (15-16).
    Step 5.2.3:let $x_i = tx^{(j)}$.
    Step 5.2.4:excute Out-of-boundary adjustment for $x_i$.
  Step 5.3:Update $\alpha$,$\beta$ and $\delta$.
  Step 5.4:let $t = t + 1$.
Step 6: return the position and fitness of $\alpha$.
```

The flow chart of the proposed algorithm is shown in Figure 1.

3.3. Analysis of hyper parameter Settings

In the proposed algorithm, two hyperparameters need to be set, which are the initial radius r and the random search dimension D of local search. The initial value of the local search radius is generally set to no more than half of the minimum value in the difference between the upper and lower bounds of all variables. If the initial search radius is too large, the individual update value will easily fluctuate, and if it is too small, the random effect of local search will not be achieved. In addition, the number of randomly selected dimensions needs to be specified. In order to ensure the randomness and representativeness of the selection, the number of selected dimensions should not be too large. Therefore, in our experiment, the number of randomly selected dimensions is selected according to Equation (17), that is, half of the data dimensions are generally selected as the random selection dimension. When the data dimension is less than or equal to 2, the random selection dimension is 1; when the data dimension is more than 12, only 6 dimensions are selected as the random selection dimension. Since the randomly selected dimension does not exceed the constant 6, the impact on the calculation time of the algorithm is only a constant times, avoiding the time cost caused by the iterative process in Powell search method.

$$D = \min(\max(1, Dim/2), 6). \tag{17}$$

Where Dim is data dimension.

3.4. Algorithm complexity analysis

This section discusses the time complexity of the proposed algorithm. Traditionally, the time complexity of swarm intelligence algorithms is a big O function of population size N, problem dimension dim and maximum number of iterations $MaxIter$. The execution time of the proposed algorithm mainly includes: population initialization, fitness evaluation, classical GWO generating individual updates, local search generating updates, combination selection, updating α,β,δ. Among them, the population initialization time complexity is $O(N)$, and the fitness evaluation takes $O(N)$. The operations of classical GWO to generate individual updates include: 6 times dim dimensional random vector generation, 3 times vector dot multiplication and difference operation and taking the absolute value, 3 times vector dot multiplication and difference operation, and 1 time to find the average value of 3 vectors, so the total time is $O(N * dim)$. The operation of local search to generate individual updates includes: randomly generate $D(D < 6)$ search directions, randomly generate D individuals, calculate the fitness value of D individuals, and select the individual with the best fitness value from the D individuals, so the total time is $D * O(N * dim)$, because $D(< 6)$ is a fixed value, so the time can still be recorded

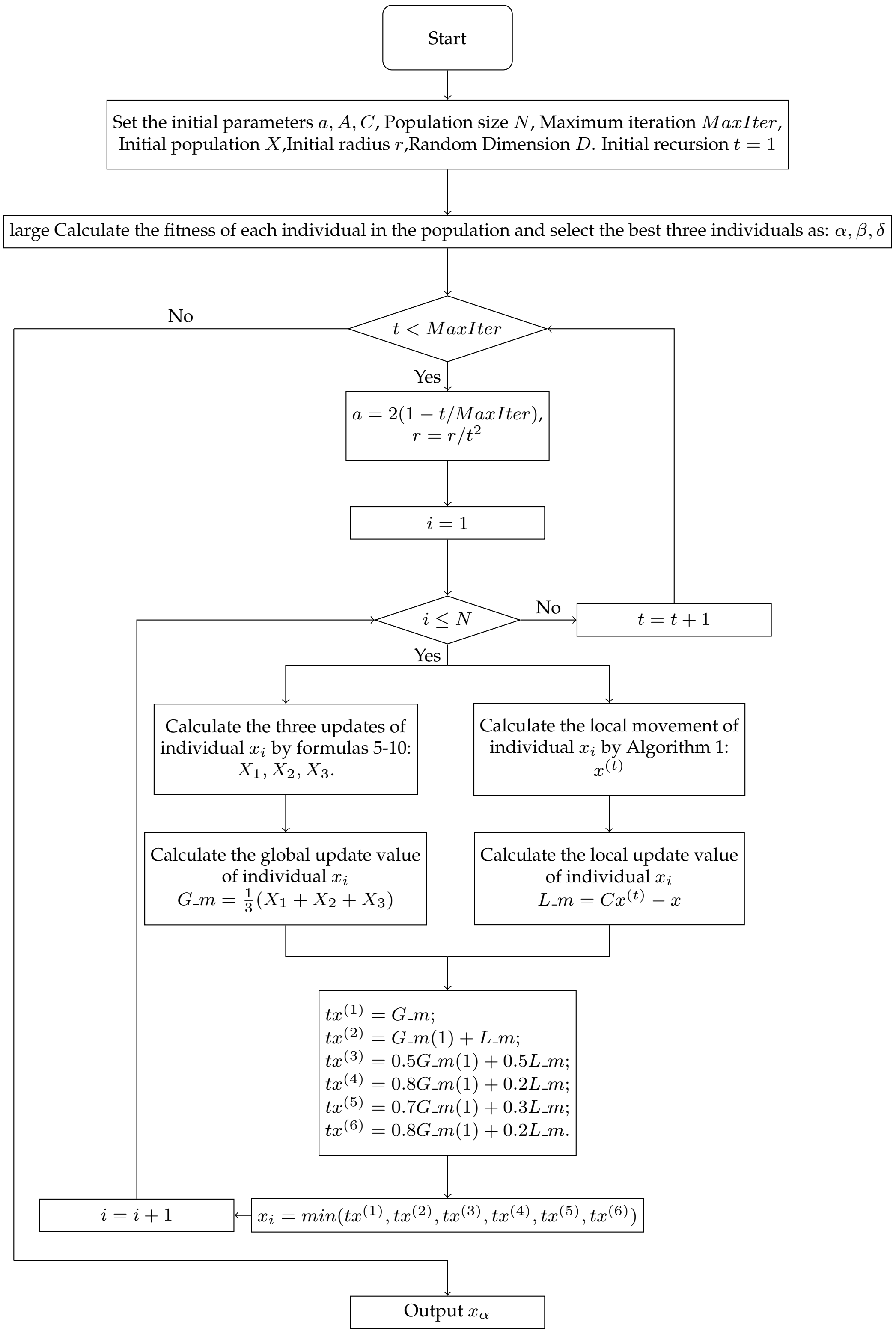

Start
Set the initial parameters a, A, C, Population size N, Maximum iteration $MaxIter$, Initial population X,Initial radius r,Random Dimension D. Initial recursion $t = 1$
large Calculate the fitness of each individual in the population and select the best three individuals as: α, β, δ
No
$t < MaxIter$
Yes
$a = 2(1 - t/MaxIter)$, $r = r/t^2$
$i = 1$
$i \leq N$
No
$t = t + 1$
Yes
Calculate the three updates of individual x_i by formulas 5-10: X_1, X_2, X_3.
Calculate the local movement of individual x_i by Algorithm 1: $x^{(t)}$
Calculate the global update value of individual x_i $G\_m = \frac{1}{3}(X_1 + X_2 + X_3)$
Calculate the local update value of individual x_i $L\_m = Cx^{(t)} - x$
$tx^{(1)} = G\_m$;
$tx^{(2)} = G\_m(1) + L\_m$;
$tx^{(3)} = 0.5G\_m(1) + 0.5L\_m$;
$tx^{(4)} = 0.8G\_m(1) + 0.2L\_m$;
$tx^{(5)} = 0.7G\_m(1) + 0.3L\_m$;
$tx^{(6)} = 0.8G\_m(1) + 0.2L\_m$.
$i = i + 1$
$x_i = min(tx^{(1)}, tx^{(2)}, tx^{(3)}, tx^{(4)}, tx^{(5)}, tx^{(6)})$
Output x_α

Figure 1: The flow chart of the presented algorithm

as $O(N * dim)$. The combination selection operation consists of generating 6 combination vectors and selecting the best vector from them with $O(N*dim)$. Updating α,β,δ can be done with $3N$ comparisons. Finally, the total time complexity of the proposed algorithm is $O(N)+MaxIter*O(N*dim)$, which can be simplified to $O(MaxIter * N * dim)$. It is on the same order of time complexity as classical GWOs.

Table 1: 23 common standard benchmark functions set.

Function No	Function name	Search range	Optimum value	Dim
f_1	Sphere	[-100,100]	$f_1(0,0,\cdots,0)=0$	30
f_2	Schwefels Problem2.22	[-10,10]	$f_2(0,0,\cdots,0)=0$	30
f_3	Schwefels Problem1.2	[-100,100]	$f_3(0,0,\cdots,0)=0$	30
f_4	Schwefels Problem2.21	[-100,100]	$f_4(0,0,\cdots,0)=0$	30
f_5	Generalized Rosenbrock	[-30,30]	$f_5(1,1,\cdots,1)=0$	30
f_6	Step	[-100,100]	$f_6(0,0,\cdots,0)=0$	30
f_7	Quartic Func-tion i.e. Noise	[-1.28,1.28]	$f_7(0,0,\cdots,0)=0$	30
f_8	Generalized Schwefel's Problem2.26	[-500,500]	$f_8(420.9687,\cdots,420.9687)$ $=-12569.5$	30
f_9	Generalized Rastrigin	[-5.12,5.12]	$f_9(0,0,\cdots,0)=0$	30
f_{10}	Ackley	[-32,32]	$f_{10}(0,0,\cdots,0)=0$	30
f_{11}	Generalized Griewank	[-600,600]	$f_{11}(0,0,\cdots,0)=0$	30
f_{12}	Generalized Penalized	[-50,50]	$f_{12}(0,0,\cdots,0)=0$	30
f_{13}	Generalized Penalized	[-50,50]	$f_{13}(0,0,\cdots,0)=0$	30
f_{14}	ShekelsFox holes	[-65.536,65.536]	$f_{14}(-32,\cdots,32)1$	2
f_{15}	Kowalik	[-5,5]	$f_{15}(\begin{smallmatrix}0.1928,0.1908,\\0.1231,0.1358\end{smallmatrix})$ ≈ 0.0003075	4
f_{16}	Six-Hump Camel-Back	[-5,5]	$min(f_{16})=-1.0316285$	2
f_{17}	Branin	[-5,0;10,15]	$min(f_{17})=0.398$	2
f_{18}	Goldstein Price	[-2,2]	$f_{18}(0,-1)=3$	2
f_{19}	Hartman Family	[0,1]	$min(f_{19})=-3.86$	3
f_{20}	Hartman Family	[0,1]	$min(f_{20})=-3.32$	6
f_{21}	Shekels Family	[0,10]	$min(f_{21})=-10.1532$	4
f_{22}	Shekels Family	[0,10]	$min(f_{22})=-10.4028$	4
f_{23}	Shekels Family	[0,10]	$min(f_{23})=-10.5363$	4

4. Experimental analysis

In order to verify the optimization performance of the proposed algorithm, 23 standard test functions for single objective optimization in literature [18], which are commonly used internationally, are selected as the test benchmark function set of this experiment, and the latest improved algorithm of G-WO is selected as the comparison benchmark algorithm. These include seven single-module functions ($f_1 \sim f_7$), six multi-module functions ($f_8 \sim f_{13}$), and ten fixed-dimensional functions ($f_{14} \sim f_{23}$). In order to facilitate comparison with other recent GWO improved algorithms, we also select a function set with 10 benchmark functions of CEC2019 used in reference [44], and 15 test function sets given in reference [47]. And 22 Real World Constraint Optimization problems from CEC2020 Real World Single Objective Constraint Optimization Competition. The all numerical experiments are run in MATLAB

R2018b on a PC with Intel(R) Core(TM) i5-9600KF 64-bit, CPU:@3.70 GHz, and 16G RAM without G-PU.The code can be download from https://www.mathworks.com/matlabcentral/fileexchange/166146-rlgwo.

4.1. Comparison of test results on a set of 23 common standard test functions

As a basic test function of single-objective optimization, 23 common standard benchmark functions are used as test functions by almost all intelligent optimization algorithms, including basic single-mode, multi-mode and fixed dimension test functions, which can reflect the basic optimization performance of an optimization algorithm. This set of functions is listed here in Table 1 for ease of reference in future research. In the experiment on this function set, in order to conveniently compare the performance of GWO algorithm improvement strategies, we choose classical GWO and the latest improved methods CMWGWO, TLGWO,LMWOAGWO and the latest SMA improved algorithm EISMA as the benchmark algorithms for comparison. In the experiment, for all algorithms, the number of population is set to 50, the maximum number of iterations is 1000, and the algorithm runs independently on each function 30 times. The experimental results are listed in Table 2.

From the test results of 23 benchmark functions listed in Table 2, it can be seen that the algorithm proposed in this paper is in a leading position in 17 of them, and the results obtained by the proposed algorithm are also very good on several other functions that are not the best results. For example, the average values of the functions f_2 and f_3 are $4.41E - 314$ and $6.92E - 259$ respectively, which are not much different from the optimal solution 0. Consider that 7 of the 23 benchmark functions are unimodal, 6 are multimodal, and 10 are fixed-dimensional. The proposed algorithm has obtained the best solution in 5 of the 7 single modular functions, and the other 2 are very close to the best solution. Therefore, the proposed algorithm has significant advantages in solving the optimization of single modular functions. On 6 multimodular functions, the proposed algorithm reaches the optimal solution in 5 of them, and only one slightly lower than the best solution of other algorithms, so it ranks second. The result shows that our algorithms are also suitable for multimode cases. Among the 10 fixed dimension benchmark functions, our algorithm achieves optimal results in 7 of them, with the optimal mean value and the smallest variance.These results show that our algorithm can also achieve good results when dealing with fixed dimension. Although EISMA, an algorithm that combines The differential evolution(DE) and slime mould algorithm(SMA), which have great influence in recent years, only reaches the optimal value on 11 benchmark functions, there is still a certain gap with the proposed algorithm.Among the other three GWO improved algorithms, only the CMWGWO reported in 2024 is optimal on the six basic functions, and the other two algorithms are lower. These are far from the proposed algorithm.It can be seen that the improvement measures of classical GWO proposed in this paper: random local search and first-element dominance can greatly improve the optimization performance of GWO.

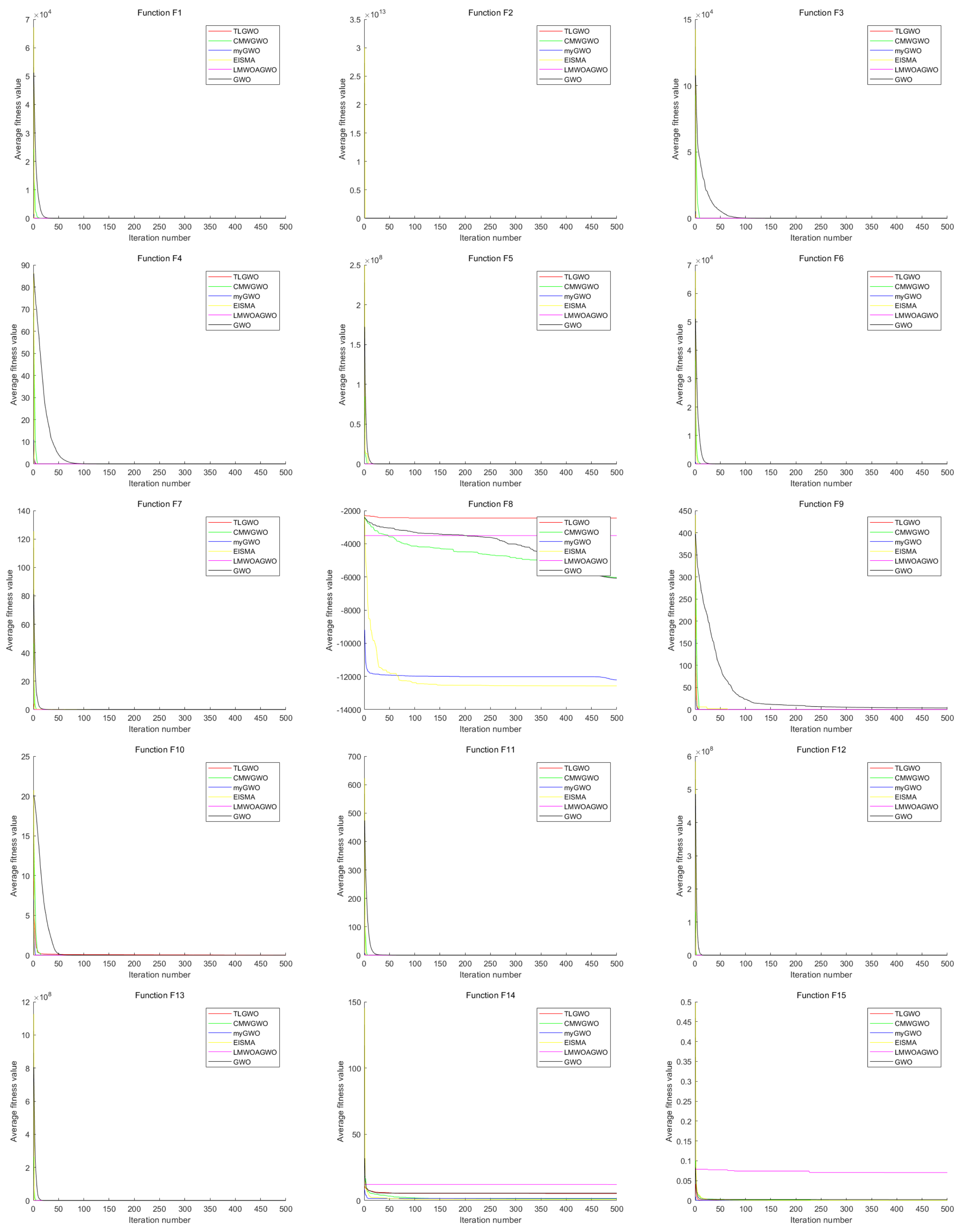
Figure 2 shows the average convergence of the execution process of 6 algorithms in 23 benchmark functions. As can be seen from Figure 2, the proposed algorithm can converge quickly on all 23 benchmark functions.Compared with the other five algorithms, the convergence speed of the proposed algorithm is one of the fastest convergence algorithms. From the convergence graphs of functions f_8, f_{14}, $f_{19} \sim f_{23}$, it can be clearly seen that the fitness of the proposed algorithm can approach the target value after very few iterations. The reason is that the proposed algorithm uses the local optimal information to optimize the grey wolf search results, and use the unique first-element advantage in the global and local combination to make the algorithm converge faster.

4.2. Comparison on 15 benchmark functions

In order to facilitate the comparison with the latest algorithm LMWOAGWO, 15 functions given in literature [47] are also cited as comparison benchmark functions, and specific function information is listed in Table 3. GWO, EISMA, CMWGWO, TLGWO, LMWOAGWO are selected as the benchmark

Table 2: Comparison of running results on 23 baseline function sets.

Function		GWO	EISMA	CMWGWO	TLGWO	LMWOAGWO	Our
f_1	Ave	3.06E-59	**0**	**0**	**0**	**0**	**0**
	(std)	(1.27E-59)	(0)	(0)	(0)	(0)	(0)
f_2	Ave	2.03E-34	**0**	**0**	**0**	4.43E-293	4.41E-314
	(std)	(5.31E-35)	(0)	(0)	(0)	(0)	(0)
f_3	Ave	2.58E-15	**0**	**0**	**0**	**0**	6.92E-259
	(std)	(1.41E-15)	(0)	(0)	(0)	(0)	(0)
f_4	Ave	1.43E-14	8.34E-216	**0**	1.73E-05	7.16E-260	**0**
	(std)	(3.95E-15)	(2.50E-214)	(0)	(1.22E-05)	(0)	(0)
f_5	Ave	26.8896	3.8322	27.3288	28.609	21.79953	**5.99E-02**
	(std)	(0.099447)	(1.6579)	(0.1268)	(0.06742)	(0)	(7.81E-03)
f_6	Ave	0.61169	0.00283	0.47281	5.2769	0.001769	**1.79E-05**
	(std)	(0.059235)	(0.00028)	(0.05835)	(0.072224)	(0.000532)	(1.03E-06)
f_7	Ave	8.89E-04	0.00014	5.6436E-05	6.69E-04	5.90E-05	**3.54E-05**
	(std)	(1.03E-04)	(1.6818E-05)	(1.232E-05)	(1.09E-04)	(4.29E-05)	(5.12E-06)
f_8	Ave	-5517.6479	**-12569.4419**	-8328.1	-2.52E03	-10207.41	-12331.0516
	(std)	(218.9043)	(0.005589)	(440.3859)	(74.0627)	(627.37695)	(161.7748)
f_9	Ave	0.31209	**0**	**0**	0	**0**	**0**
	(std)	(0.19578)	(0)	(0)	(0)	(0)	(0)
f_{10}	Ave	1.64E-14	8.8818E-16	8.8818E-16	9.88E-03	**4.44E-16**	8.88E-16
	(std)	(4.88E-16)	(0)	(0)	(1.56E-03)	(0)	(0)
f_{11}	Ave	2.63E-03	**0**	**0**	**0**	**0**	**0**
	(std)	(1.12E-03)	(0)	(0)	(0)	(0)	(0)
f_{12}	Ave	0.38637	0.00178	0.031757	0.69952	0.00293	**2.69E-06**
	(std)	(3.22E-03)	(0.00041)	(0.0028)	(0.028411)	(0.001196)	(1.99E-07)
f_{13}	Ave	0.49832	0.00364	0.8404	2.98	0.001169	**4.13E-05**
	(std)	(0.035318)	(0.00052)	(0.095297)	(5.22E-03)	(0.000348)	(2.49E-06)
f_{14}	Ave	4.9755	**0.998**	1.1634	4.9573	2.25442	2.0736
	(std)	(0.81268)	(1.099E-11)	(0.09452)	(0.71187)	(1.00901)	(0.53585)
f_{15}	Ave	3.04E-03	0.000393	0.0004657	5.01E-04	0.000308	**3.07E-04**
	(std)	(1.24E-03)	(1.9839E-05)	(2.4076E-05)	(2.69E-05)	(1.08E-08)	(4.41E-09)
f_{16}	Ave	**-1.0316**	**-1.0316**	-1.0315	-1.0301	-1.031628	**-1.0316**
	(std)	(6.1574E-10)	(9.402E-09)	(1.7844E-05)	(1.08E-03)	(2.76E-07)	(4.329E-10)
f_{17}	Ave	**0.39789**	**0.39789**	0.39843	0.39859	0.397897	**0.39789**
	(std)	(2.41E-08)	(1.2E-07)	(0.000384)	(1.33E-04)	(7.95E-06)	(9.32E-09)
f_{18}	Ave	**3**	**3**	3.00	3.0001	3.000003	**3**
	(std)	(1.29E-06)	(4.488E-09)	(4.8208E-06)	(4.07E-05)	(4.99E-05)	(2.23E-07)
f_{19}	Ave	-3.8619	**-3.8628**	-3.8628	-3.8547	-3.86277	-3.8622
	(std)	(4.19E-04)	(4.133E-06)	(6.563E-06)	(8.76E-04)	(5.57E-06)	(3.43E-04)
f_{20}	Ave	-3.22	-3.2305	-3.2548	-3.0498	-3.29415	**-3.322**
	(std)	(2.93E-07)	(0.00921)	(0.011)	(2.04E-02)	(0.5113)	(7.66E-07)
f_{21}	Ave	-9.8162	-10.153	-10.074	-6.0234	-10.1421	**-10.1531**
	(std)	(0.2301)	(2.825E-05)	(0.0737)	(0.25356)	(0.007254)	(1.72E-05)
f_{22}	Ave	-10.4027	-10.4027	-10.2734	-6.0858	-10.3998	**-10.4028**
	(std)	(1.72E-05)	(5.2185E-05)	(0.1165)	(0.24856)	(0.008346)	(1.68E-05)
f_{23}	Ave	-10.5362	-10.5362	-10.4057	-6.2545	-10.5269	**-10.5363**
	(std)	(1.10E-05)	(6.529E-05)	(0.0894)	(0.18648)	(0.007653)	(1.791E-05)



Function F1
Function F2
Function F3
Function F4
Function F5
Function F6
Function F7
Function F8
Function F9
Function F10
Function F11
Function F12
Function F13
Function F14
Function F15
TLGWO
CMWGWO
myGWO
EISMA
LMWOAGWO
GWO
Average fitness value
Iteration number

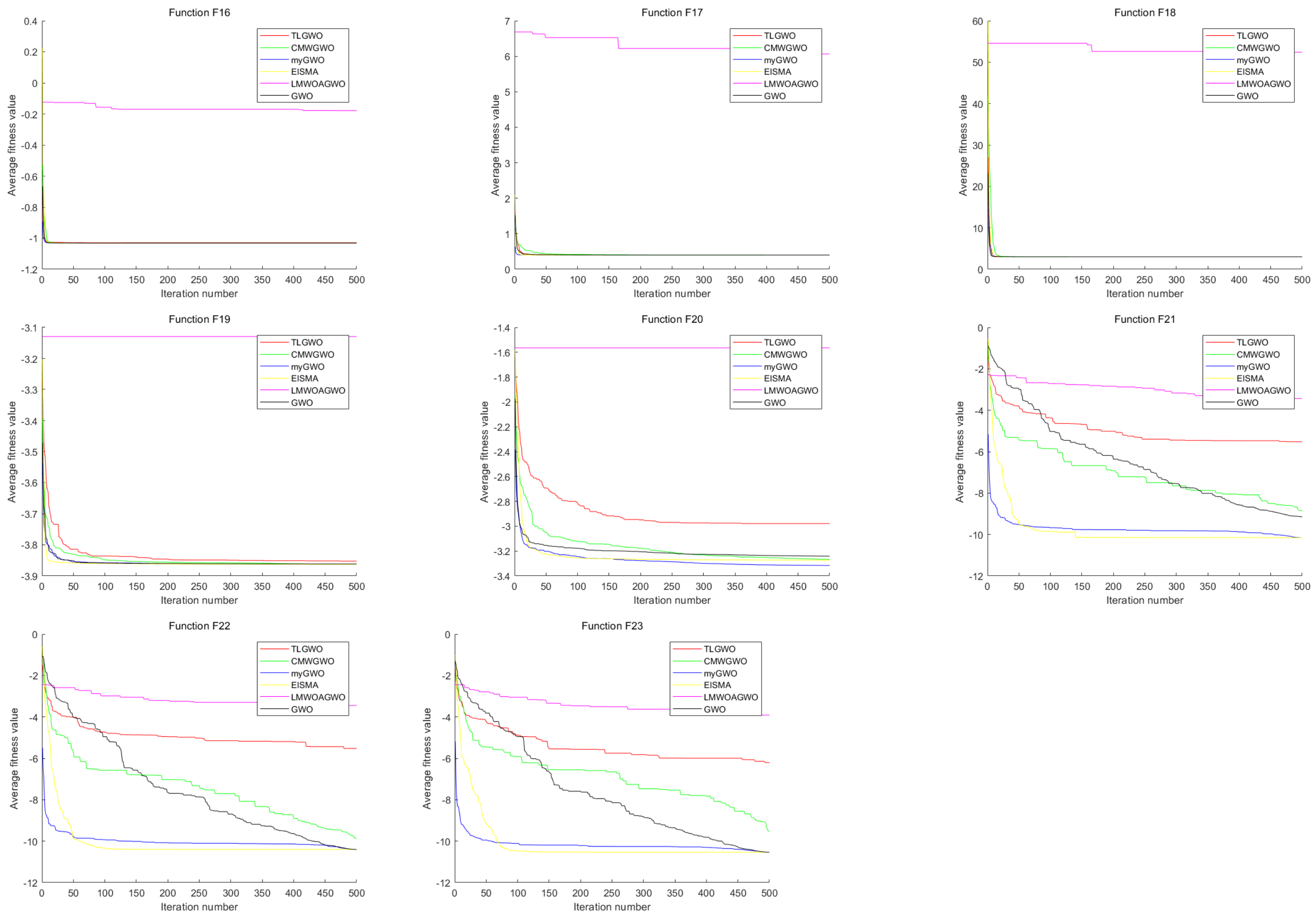


Figure 2: Comparison of convergence of 6 algorithms on 23 benchmark functions.

algorithms for comparison. In our experiment, the population size was set to 50, the recursion number to 1000, and the data dimension was unified to 30. Each algorithm was independently repeated 30 times on each function, and its average and variance were counted. The results were listed in Table 4. It can be seen from Table 4 that among the 15 benchmark functions, the average value of the proposed algorithm achieves the optimal solution in 12 of them. The optimal solution is not reached in the other three functions, the error rate between the result of the proposed algorithm and the optimal solution does not exceed 2E-05. Especially for the function f_4, the optimization results obtained by the proposed algorithm have obvious advantages over other three GWO improved algorithms. It can be seen that the proposed improvement measures has obvious superior performance.

Table 3: 15 test benchmark functions[45].

Function Name	Boundaries	$f(x^*)$	Type
f_1:Sphere	[-100,100]	0	unimodal
f_2:Schwefels Problem 2.22	[-10,10]	0	unimodal
f_3:Step	[-100,100]	0	unimodal
f_4:Rosenbrock	[-2.048,2.048]	0	multimodal
f_5:Rotated Hyper-ellipsoid	[-100,100]	0	unimodal
f_6:Schwefels problem 2.26	[-500,500]	$-418.98 * d$	multimodal
f_7:Rastrigin	[-5.12,5.12]	0	multimodal
f_8:Ackley	[-32,32]	0	multimodal
f_9:Griewank	[-600,600]	0	multimodal
f_{10}:Six-Hump Camel-Back	[-5,5]	-1.031628	multimodal
f_{11}:Shifted Sphere	[-100,100]	-450	unimodal
f_{12}:Shifted Schwefels problem 1.2	[-100,100]	-450	unimodal
f_{13}:Shifted Rosenbrock	[-100,100]	390	multimodal
f_{14}:Shifted Rastrigin	[-5,5]	-330	multimodal
f_{15}:Shifted Expanded Griewanks plus Rosenbrocks Function	[-5,5]	-130	multimodal

4.3. Comparison on 10 benchmark functions of CEC2019

In order to facilitate the performance comparison with the improved methods of the latest GWO algorithm, the single objective optimization test benchmark function in CEC2019 adopted by LMWOAGWO algorithm is also selected as the benchmark function set for comparing the performance of each optimization algorithm. The algorithm used for comparison is the same as before, and the population size is still set to 50, the recursion number is 1000, and the data dimension is unified to 30. Each algorithm is independently repeated 50 times on each function, and its average and variance are counted. The results are listed in Table 5. It can be seen from Table 5 that EISMA has obtained the optimal solution in 7 of the 10 benchmark functions. However, The results obtained by the proposed algorithm are not significantly different from those obtained by EISMA, especially from a practical point of view. Compared with CMWGWO algorithm, the results of the proposed algorithm are better in 9 of the 10 benchmark functions, and the results of our algorithm are also better than TLGWO on these 10 functions. Compared with LMWOAGWO algorithm, the proposed algorithm is performs on 7 functions, and the average fitness of LMWOAGWO on f_1 function is more than 10^4 times that of the proposed algorithm, indicating that The proposed improvement measures are very suitable for GWO.

4.4. Comparison of 22 real application constraint optimization problems

To facilitate comparison and alignment with the benchmarks used in the literature [44], we cite the 22 constraint optimization problems selected from ”CEC2020 Real World Single Objective Constraint

Table 4: Comparison of execution results on 15 benchmark functions set.

Function		GWO	EISMA	CMWGWO	TLGWO	LMWOAGWO	Our
f_1	Ave	1.39E-69	**0**	**0**	**0**	**0**	**0**
	(std)	(5.88E-69)	(0)	(0)	(0)	(N/A)	(0)
f_2	Ave	6.89E-41	1.56E-220	**0**	5.12E-09	**0**	**0**
	(std)	(1.01E-40)	(0)	(0)	(5.03E-09)	(N/A)	(0)
f_3	Ave	**0**	**0**	**0**	**0**	4.05E-05	**0**
	(std)	(0)	(0)	(0)	(0)	(N/A)	(0)
f_4	Ave	26.2846	0.085	26.9571	28.616	6.86	**2.81E-02**
	(std)	(0.70208)	(0.0802)	(0.6047)	(5.85E-02)	(N/A)	(3.09E-03)
f_5	Ave	3.24E-18	**0**	**0**	**0**	**0**	**0**
	(std)	(1.46E-17)	(0)	(0)	(0)	(N/A)	(0)
f_6	Ave	-5710.2323	**-12569.46**	-6952.8236	-2520.0731	-4.06E+03	-12450.2777
	(std)	(1259.6292)	(0.0184)	(1583.498)	(66.7175)	(N/A)	(11.4327)
f_7	Ave	0.73146	**0**	**0**	1.34E-04	**0**	**0**
	(std)	(2.0617)	(0)	(0)	(4.29E-05)	(N/A)	(0)
f_8	Ave	1.18E-14	**0**	**0**	0.866E-03	4.44E-16	**0**
	(std)	(3.14E-15)	(0)	(0)	(1.12E-03)	(N/A)	(0)
f_9	Ave	2.04E-03	**0**	**0**	**0**	**0**	**0**
	(std)	5.01E-03	(0)	(0)	(0)	(N/A)	(0)
f_{10}	Ave	-1.0316	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**
	(std)	(1.51E-09)	(2.87E-09)	(4.676E-05)	(6.69E-06)	(N/A)	(1.26E-10)
f_{11}	Ave	**-450**	**-450**	-449.9999	-449.5921	**-4.50E+02**	**-450**
	(std)	(6.97E-06)	(1.01E-06)	(6.0387E-05)	(6.09E-02)	(N/A)	(1.13E-06)
f_{12}	Ave	**-450**	**-450**	-449.9999	-449.4666	5.99E+02	**-450**
	(std)	(5.34E-06)	(5.2502E-07)	(0.00014)	(8.11E-02)	(N/A)	(4.39E-06)
f_{13}	Ave	-378.5162	**-390**	-385.0514	-359.2712	7.77E+02	-389.9996
	(std)	(16.1641)	(2.1345E-06)	(12.8393)	(2.841)	(N/A)	(5.46E-04)
f_{14}	Ave	-329.9337	**-330**	**-330**	-329.9505	-3.29E+02	**-330**
	(std)	(0.25243)	(2.6766E-06)	(5.411E-05)	(8.55E-03)	(N/A)	(1.11E-06)
f_{15}	Ave	-129.994	-129.9987	**-130**	-129.9754	-1.29E+02	-129.9974
	(std)	(8.49E-02)	(0.0034)	(9.5773E-12)	(5.61E-03)	(N/A)	(5.77E-03)

Optimization Competition" in literature [44]. In order to facilitate subsequent reference and comparison of algorithm results, these 22 questions are also listed in Table 6. In the experiment, the total population number is still set to 50, the recursion number to 1000, each algorithm is independently repeated 30 times on each function. The local search radius and random search dimension are set as before. Assuming that the default tolerance for equality constraints is 1.0E-4, consistent with the competition setting. Considering the constraint conditions in constraint optimization, we make new adjustments to population initialization and transboundary processing in the proposed algorithm.

4.4.1. Initial population generation

Considering the difference between constrained optimization and unconstrained optimization, a new initial population generation method is proposed. The method can satisfies the constraint optimization requirements, ensures that each individual of the population meets the constraint requirements before the optimization solution, and also considers that the optimal solution may appear at the boundary of the search domain. The specific generation idea is as follows: Firstly, the population is ran-

Table 5: Comparison of running results on 10 benchmark function sets in CEC2019.

Function		GWO	EISMA	CMWGWO	TLGWO	LMWOAGWO	Our
f_1	Ave	9932.3821	**1**	**1**	**1**	4.60E+04	**1**
	(std)	(7540.9519)	(0)	(0)	(0)	(7.19E+03)	(9.34E-15)
f_2	Ave	253.2205	4.9174	4.6591	4.9573	17.354	**4.3177**
	(std)	(21.7674)	(0.0279)	(0.05225)	(2.35E-02)	(0.014)	(0.028573)
f_3	Ave	2.4153	**2.012**	5.6013	6.5156	12.702	3.6751
	(std)	(0.23005)	(0.20536)	(0.14955)	(0.13779)	(4.03E-07)	(0.18646)
f_4	Ave	15.5776	**11.5934**	23.4209	64.1722	57.726	14.9788
	(std)	(1.1126)	(0.52911)	(1.945)	(1.2791)	(10.991)	(0.95186)
f_5	Ave	1.6135	**1.1471**	1.917	16.7454	1.187	1.6449
	(std)	(0.084049)	(0.01068)	(0.0077374)	(1.2994)	(0.087)	(0.093216)
f_6	Ave	2.4578	3.3948	2.3051	7.2895	6.255	**2.2332**
	(std)	(0.14473)	(0.029763)	(0.13643)	(0.15141)	(2.063)	(0.13705)
f_7	Ave	646.148	381.9556	954.49	1769.2119	**202.898**	645.3191
	(std)	(42.2273)	(26.7748)	(64.769)	(39.0005)	(128.763)	(19.2943)
f_8	Ave	3.6179	**3.3485**	3.6569	4.9401	4.991	3.485
	(std)	(0.077472)	(0.05451)	(0.066094)	(0.040767)	(0.733)	(0.077868)
f_9	Ave	1.1579	**1.1215**	1.2201	1.6215	3.208	1.1577
	(std)	(0.0068281)	(0.004166)	(0.010121)	(0.040381)	(0.453)	(0.010406)
f_{10}	Ave	20.9886	**17.4703**	19.686	21.3389	19.781	20.9752
	(std)	(0.46017)	(1.0806)	(0.70965)	(0.01192)	(3.320)	(0.1872)

domly generated according to the normal population initialization method, and then detect whether there are individuals meeting the constraint requirements in the generated population. If it exists, the population individuals meeting the constraint requirements are saved, then check whether the number of populations meeting the requirements reaches the set value, and the initialization ends when it reaches the set value. Otherwise, new populations are randomly generated and the constraint satisfaction of the generated population individuals is tested. If there are no individuals in the generated population that meet the constraint requirements, turn to generating individuals that may be located at the boundary of the search domain. Individuals at the boundaries of the search domain are divided into three categories: mixed individuals with upper and lower bounds, individuals with only upper bounds, and individuals with only lower bounds. Mixed population individual generation method: The population is generated randomly, and then some individuals are randomly selected to produce mixed boundary population, and the rest are used to produce unilateral individuals. Some dimensions are randomly selected from the ones used to generate mixed population individuals to assign the upper bound, and then some dimensions are randomly selected from the remaining dimensions to assign the lower bound. After the mixed boundary population is generated, it is randomly determined whether other population individuals are used to generate upper boundary population individuals or lower boundary population individuals. By randomly selecting some dimensions from the individual dimension and assigning corresponding boundaries to the search domain,both the upper boundary individual and the lower boundary individual are generated. The specific description is given by the pseudo-code in algorithm 3.

Algorithm 3. An initial population generation method with boundary that satisfies the constraint conditions.

Input: population size N, Upper bound ub and lower bound lb, Data dimension Dim, Problem No. f_num, The default tolerance of the equation constrains ϵ.

Table 6: 22 Real-world constrained optimization problem[44].

Problem No	Problem Name	Optimism value $f(x^{*})$
RW01	Process synthesis problem	2
RW02	Process synthesis and design problem	2.557654574
RW03	Process flow sheeting problem	1.076543083
RW04	Process design Problem	26887
RW05	Multi-product batch plant	53638.94272
RW06	Weight Minimization of a Speed Reducer	2994.424466
RW07	Optimal Design of Industrial refrigeration System	0.032213001
RW08	Tension/compression spring design (case 1)	0.012665233
RW09	Pressure vessel design	5885.332774
RW10	Welded beam design	1.670217726
RW11	Three-bar truss design problem	263.8958434
RW12	Multiple disk clutch brake design problem	0.235242458
RW13	Planetary gear train design optimization problem	0.525768707
RW14	Step-cone pulley problem	16.06986873
RW15	Hydro-static thrust bearing design problem	1625.442809
RW16	10-bar truss design	524.4507607
RW17	Rolling element bearing	14614.13572
RW18	Gas Transmission Compressor Design (GTCD)	2964895.417
RW19	Tension/compression spring design (case 2)	2.613884058
RW20	Gear train design Problem	0
RW21	Himmelblaus Function	-30665.53867
RW22	Topology Optimization	2.639346497

Output: Initial population.
Let $tnum = 0$.
Execute the following loop when $tnum < N$:

(1) Randomly generate a set of individuals xt of size N, Then calculate the fitness f, the inequality constraint equation value g and the equality constraint equation value h for each individual.

(2) Count the number of individuals meeting the constraint conditions tn.

(3) If $tn < 1$, then use the following procedure to generate an individual at the boundary.

(3.1) Randomly generate the individual index set ptp for generating mixed boundaries, and the single boundary individual index set spp;

(3.2) Randomly generate the dimension sets that reach the upper and lower boundaries $ub0$ and $lb0$;

(3.3) Let $xt\,(ptp, ub0) = ub(ub0)$, $xt\,(ptp, lb0) = lb(lb0)$;

(3.4) If $rand > 0.5$, then

Randomly generated the upper boundary dimension set $ub0$.

$xt\,(spp, ub0) = ub(ub0)$.

else

Randomly generated the lower boundary dimension set $lb0$.

$xt\,(spp, lb0) = lb(lb0)$.

(3.5) Calculate the fitness f, the value of the inequality constraint equation g and the value of the equality constraint equation h of the individual set xt.

(3.6) Count the number of individuals meeting the constraint conditions tn.

(4) If $tn > 0$

(4.1)The individuals meeting the constraint conditions are selected and sorted according to the ascending order of fitness.

(4.2) The number of individuals that can be selected as a population is calculated using the following formula:

$gtn = min(N - tnum, tn)$.

(4.3) The first gtn individuals meeting the constraint conditions were selected to join the population.

(4.4) $tnum = tnum + gtn$.

(5)Output initial population.

4.4.2. Transboundary detection and processing

In constraint optimization, in addition to detecting whether the newly generated individuals are beyond the boundary of the search domain, it is also necessary to check whether they meet the constraint requirements. The general transboundary processing method that only intercepts the transboundary dimension as the boundary is obviously unable to meet the constraint optimization requirements. Therefore, this paper proposes a new transboundary treatment method, which focuses on ensuring that the optimal movement direction of each individual in the population remains unchanged during the solution process. For individual x in the population, assume that the optimized new position coordinate is $x1$. Considering that each individual is guaranteed to meet the constraints when the population is initialized. Therefore, we can search gradually from $x1$ to x along the line of x and $x1$ until there is a point that satisfies the constraints, and select that point as the new optimized position of individual x. The specific calculation process is shown in algorithm 4.

Algorithm 4. Transboundary processing.

Input: Individual x and its updated position $x1$, upper bound ub and lower bound lb.

Output: New coordinate value.

```
(1) The dimension value of $x1$ that exceeds the boundary of the search domain is taken as the boundary value.
(2) Let $dx = x1 - x; t = 0; tx = x1$.
(3) If $tx$ does not satisfy the constraint, the following procedure is repeated:
    (3.1) $tx = x + \frac{1}{e^t}dx$.
    (3.2) $t = t + 1$.
(4) Output $tx$.
```

In the specific application, in order to ensure that the number of cycles in algorithm 4 is not excessive, the number of loops can be limited by t, If the constraint is still not satisfied after the set number of loops tx,let the sequence exit,$tx = x$.

4.4.3. Comparison and analysis

In order to facilitate comparison, we directly cite the experimental results of literature[44] and attach the experimental results of the algorithm proposed in this paper to the last column of the original table, forming Table 7. Considering that the process of constraint optimization requires the processing of constraint violation terms, the relevant experimental data are not provided in the literature of the algorithm EISMA and CMWGWO, so they are not involved in the comparison here. It can be seen from the data in the table that in 13 of the 22 practical problems, the proposed algorithm has obtained the optimal solution. It shows that the proposed algorithm has a strong competition in solving practical constrained optimization problems, and in many problems our algorithm reaches or approximates to the ideal optimal solution. The reasons can be summarized into three points: First, the algorithm considers the influence of global optimization and local optimization on the optimization process; Secondly, The first-element dominance can greatly improve the efficiency of global and local information integration. Third,each individual in the generated initial population satisfies the constraint requirements, which can alleviate the instability caused by the deviation from processing that the constraint violation in the subsequent optimization process. Finally, the new transboundary processing strategy guarantees the optimal direction under the premise of satisfying the constraint requirements.

4.5. Comparative analysis of first-element advantage

In order to verify the effect of using first-element advantage, we perform an experimental comparison between using first-element dominance and not using first-element dominance on 23 general test benchmark function sets. The experimental results are listed in Table 8. In Table 8, C-GWO represents classical GWO, fe-GWO (algorithm proposed in this paper) and nfe-GWO represent local information-optimized GWO with and without a first-element dominance, respectively. As can be seen from Table 8, local information optimized GWO with first-element dominance has obvious advantages. In order to clearly compare the influence of first-element dominance on the convergence rate, three function are selected as the representative from single module, multi-module and fixed dimension, which are respectively f_1, f_8 and f_{14}. The average fitness variation trend in the recursive process on these three functions are shown in Figure 3. It can be seen from Figure 3 that the first-element advantage can significantly accelerate algorithm convergence.

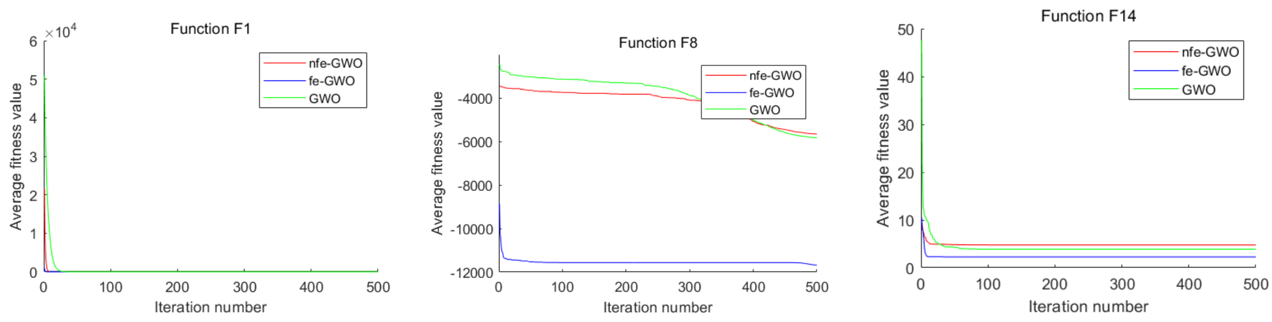


Figure 3: Comparison of GWO convergence between local information optimization with first element dominance and without first element dominance.

5. Summary

Grey Wolf algorithm attracts many researchers to study it because of its simplicity, high efficiency and wide application prospect. In order to solve the problem that the classical GWO algorithm focus on global information and ignore local information, a new method to modify GWO results by using local information is proposed. Compared with other methods using local information to optimize G-WO algorithm, our proposed combination of local information and GWO search results is simpler and more efficient, and a special first-element dominance is used in the combination to greatly improve the performance of the algorithm. In the local optimization process, we use the random positioning method near the fixed position with multi-dimensional directions, which reduces the algorithm complexity and preserves the randomness of the positioning as much as possible. In order to apply GWO to constrained optimization, a hybrid initial population initialization method is proposed to ensure that all population individuals meet the constraint requirements at the beginning of the algorithm. A new method is used to search for solutions that meet the constraints in the update direction to ensure that the constraints are not violated in the solution process, instead of the traditional truncation and cross-processing methods. Experimental results verify the effectiveness of the proposed method. But what is the theoretical basis of first -element dominance and its adaptive range still need to be further studied.

Table 7: Comparison of 22 real application optimization problems in CEC2020.

Function		iLSHADEϵ	sCMAgES	COLSHADE	EnMODE	LMWOAGWO	Our
RW01	Ave	**2.00E + 00**	1.99E + 00	**2.00E + 00**	**2.00E + 00**	**2.00E + 00**	**2**
	(std)	(0.00E + 00)	(1.52E - 01)	(0.00E + 00)	(0.00E + 00)	(1.30E - 07)	(0)
RW02	Ave	2.56E + 00	2.55E + 00	2.56E + 00	2.56E + 00	2.56E + 00	**2.5577**
	(std)	(1.46E - 05)	(2.70E - 01)	(0.00E + 00)	(1.36E - 15)	(2.17E - 05)	(8.59E - 08)
RW03	Ave	1.22E + 00	1.08E + 00	1.10E + 00	1.15E + 00	1.19E + 00	**1.0765**
	(std)	(6.48E - 02)	(3.47E - 02)	(6.36E - 02)	(8.79E - 02)	(8.53E - 02)	(6.53E - 08)
RW04	Ave	2.69E + 04	2.69E + 04	2.69E + 04	2.69E + 04	2.69E + 04	**26887.4414**
	(std)	(1.11E - 11)	(1.11E - 11)	(3.64E - 12)	(1.11E - 11)	(7.50E - 03)	(3.71E-03)
RW05	Ave	5.91E + 04	5.81E + 04	5.85E + 04	5.85E + 04	**5.47E + 04**	58569.0977
	(std)	(1.85E + 03)	(1.35E + 03)	(7.28E - 12)	(8.06E - 09)	(2.45E + 03)	(7.1233)
RW06	Ave	**2.99E + 03**	**2.99E + 03**	**2.99E + 03**	**2.99E + 03**	3.00E + 03	3003.2568
	(std)	(4.64E - 13)	(4.64E - 13)	(4.55E - 13)	(4.64E - 13)	(9.40E - 01)	(0.6903)
RW07	Ave	3.82E-01	**3.22E-02**	**3.22E-02**	**3.22E-02**	4.01E-02	0.036561
	(std)	(9.67E - 01)	(2.78E - 17)	(0.00E + 00)	(3.17E - 18)	(4.43E - 03)	(3.13E-04)
RW08	Ave	1.30E-02	1.27E-02	1.27E-02	1.27E-02	1.27E-02	**0.012667**
	(std)	(1.06E - 03)	(2.16E - 04)	(1.06E - 07)	(2.01E - 05)	(5.32E-05)	(8.22E-07)
RW09	Ave	8.48E + 03	7.38E + 03	**6.06E + 03**	**6.06E + 03**	6.58E + 03	6142.9897
	(std)	(3.14E + 03)	(1.93E + 03)	(8.36E + 00)	(9.28E - 13)	(4.58E + 02)	(3.61E+01)
RW10	Ave	**1.67E + 00**	1.69E + 00	**1.67E + 00**	**1.67E + 00**	**1.67E + 00**	**1.6700**
	(std)	(7.59E - 07)	(3.95E - 02)	(0.00E + 00)	(0.00E + 00)	(4.03E-04)	(3.10E-06)
RW11	Ave	2.64E + 02	2.65E + 02	2.64E + 02	2.64E + 02	2.64E + 02	**263.8959**
	(std)	(1.99E - 02)	(2.88E + 00)	(0.00E + 00)	(0.00E + 00)	(1.02E - 04)	(1.04E-05)
RW12	Ave	2.35E-01	2.35E-01	2.35E-01	2.35E-01	2.35E-01	**0.23525**
	(std)	(1.13E - 16)	(1.13E - 16)	(0.00E + 00)	(1.13E - 16)	(1.68E - 07)	(1.15E-06)
RW13	Ave	5.27E-01	6.16E-01	5.41E-01	5.27E-01	**5.26E-01**	0.52716
	(std)	(1.69E - 03)	(1.98E - 01)	(4.26E - 02)	(1.44E - 03)	(5.60E - 04)	(3.03E-04)
RW14	Ave	1.61E + 01	1.61E + 01	1.61E + 01	1.61E + 01	1.65E + 01	**16.0842**
	(std)	(8.62E - 08)	(1.78E - 14)	(0.00E + 00)	(3.33E - 14)	(2.09E - 01)	(0.098437)
RW15	Ave	1.76E + 03	2.35E + 03	1.64E + 03	**1.62E + 03**	1.83E + 03	1839.3224
	(std)	(1.28E + 03)	(1.41E + 03)	(1.01E + 02)	(1.78E - 11)	(5.52E + 01)	(1.32E+01)
RW16	Ave	5.25E + 02	5.30E + 02	**5.24E + 02**	**5.24E + 02**	5.29E + 02	530.4093
	(std)	(7.14E - 02)	(2.03E + 00)	(0.00E + 00)	(3.76E - 07)	(2.32E + 00)	(0.20647)
RW17	Ave	1.46E + 04	1.46E + 04	1.70E + 04	1.70E + 04	1.70E + 04	**14635.9492**
	(std)	(9.28E - 12)	(9.28E - 12)	(0.00E + 00)	(3.71E - 12)	(9.29E + 00)	(2.8878)
RW18	Ave	2.97E + 06	2.96E + 06	2.96E + 06	2.96E + 06	2.96E + 06	**2964906.5**
	(std)	(6.57E + 02)	(1.43E - 09)	(0.00E + 00)	(1.43E - 09)	(6.15E + 00)	(1.8808)
RW19	Ave	6.26E + 00	**2.21E + 00**	2.66E + 00	2.81E + 00	2.71E + 00	2.6586
	(std)	(6.32E + 00)	(1.22E + 00)	(1.11E - 02)	(3.66E - 01)	(9.48E - 02)	(0)
RW20	Ave	5.56E-17	**0.00E + 00**	1.88E-16	**0.00E + 00**	9.53E-18	2.90E-13
	(std)	(1.17E - 16)	(0.00E + 00)	(3.81E - 16)	(0.00E + 00)	(1.54E - 17)	(1.27E-13)
RW21	Ave	-3.07E + 04	-3.07E+04	-3.07E + 04	-3.07E + 04	-3.07E + 04	**-30663.541**
	(std)	(3.64E - 12)	(3.56E - 12)	(0.00E + 00)	(3.71E - 12)	(3.18E-01)	(2.01E-01)
RW22	Ave	2.64E + 00	2.65E + 00	2.64E + 00	2.64E + 00	2.64E + 00	**2.6393**
	(std)	(8.11E - 16)	(1.26E - 02)	(0.00E + 00)	(1.02E - 15)	(1.36E - 15)	(0)

Table 8: Comparative of whether and or first-element advantage.

	f_1	f_2	f_3	f_4	f_5	f_6
C-GWO	3.06E-59	2.03E-34	2.58E-15	1.43E-14	26.8896	0.61169
	(1.27E-59)	(5.31E-35)	(1.41E-15)	(3.95E-15)	(0.099447)	(0.059235)
nfe-GWO	3.9477E-296	2.8679E-151	7.1774E-96	6.8032E-129	26.3821	3.9952
	(0)	(3.3336E-152)	(7.0488E-96)	(1.0304E-129)	(0.11907)	(0.18475)
fe-GWO	**0**	**4.41E-314**	**6.92E-259**	**0**	**5.99E-02**	**1.79E-05**
	(0)	(0)	(0)	(0)	(7.81E-03)	(1.03E-06)

	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
C-GWO	8.89E-04	-5517.6479	0.31209	1.64E-14	2.63E-03	0.38637
	(1.03E-04)	(218.9043)	(0.19578)	(4.88E-16)	(1.12E-03)	(3.22E-03)
nfe-GWO	9.2298E-05	-5474.704	0	6.9278E-15	0.00041109	0.21114
	(1.5026E-05)	(212.7478)	(0)	(2.9724E-16)	(0.00040418)	(0.040277)
fe-GWO	**3.54E-05**	**-12331.0516**	**0**	**8.88E-16**	**0**	**2.69E-06**
	(5.12E-06)	(161.7748)	(0)	(0)	(0)	(1.99E-07)

	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
C-GWO	0.49832	4.9755	3.04E-03	-1.0316	0.39789	3
	(0.035318)	(0.81268)	(1.24E-03)	(6.1574E-10)	(2.41E-08)	(1.29E-06)
nfe-GWO	0.2072	4.5306	0.0003075	-1.0316	0.39789	3
	(0.022406)	(0.66925)	(4.5014E-09)	(5.2578E-10)	(1.2917E-08)	(5.4895E-07)
fe-GWO	**4.13E-05**	**2.0736**	**3.07E-04**	**-1.0316**	**0.39789**	**3**
	(2.49E-06)	(0.53585)	(4.41E-09)	(4.329E-10)	(9.32E-09)	(2.23E-06)

	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	
C-GWO	-3.8619	-3.22	-9.8162	-10.4027	-10.5362	
	(4.19E-04)	(2.93E-07)	(0.2301)	(1.72E-05)	(1.10E-05)	
nfe-GWO	**-3.8628**	**-3.322**	**-10.1531**	**-10.4028**	**-10.5363**	
	(6.8696E-06)	(4.8124E-07)	(1.0303E-05)	(1.2999E-05)	(1.1767E-05)	
fe-GWO	-3.8622	**-3.322**	**-10.1531**	**-10.4028**	**-10.5363**	
	(3.43E-04)	(7.66E-07)	(1.72E-05)	(1.68E-05)	(1.791E-05)	

6. ACKNOWLEDGEMENTS

This work was financially supported by the National Natural Science Foundation of China (62062011, 61866003).

References

[1] Minzu, V., Serbencu, A.. Systematic Procedure for Optimal Controller Implementation Using Metaheuristic Algorithms. Intelligent Automation & Soft Computing, no.26, pp.663-677,2020.

[2]Castillo, O., Melin, P.. A Review of Fuzzy Metaheuristics for Optimal Design of Fuzzy Controllers in Mobile Robotics. In Complex Systems: Spanning Control and Computational Cybernetics: Applications: Dedicated to Professor Georgi M. Dimirovski on His Anniversary; Shi,P., Stefanovski, J., Kacprzyk, J., Eds.; Springer International Publishing: Cham, Switzerland, pp.59-72,2022.

[3]Razmjooy, N., Ashourian, M., Foroozandeh, Z.. Metaheuristics and Optimization in Computer and Electrical Engineering, Springer Nature Switzerland AG: Cham, Switzerland,2021.

[4]Vineeth, P., Suresh, S.. Performance evaluation and analysis of population-based metaheuristics for denoising of biomedical images. Research on Biomedical Engineering, no.37, pp.111-133,2021.

[5]Nssibi, M., Manita, G., Korbaa, O.. Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey. Computer Science Review, vol.49,2023,https://doi.org/10.1016/j.cosrev.2023.100559..

[6]Aslay, S.E., Dede, T.. Reduce the construction cost of a 7-story RC public building with metaheuristic algorithms. Architectural Engineering and Design Management, pp.1-16,2023.

[7]Amorim, A.R., Zafalon, G.F.D., Contessoto, A.d.G., Valêncio, C.R., Sato, L.M.. Metaheuristics for multiple sequence alignment: A systematic review. Computational Biology and Chemistry, vol.94,2021,https://doi.org/10.1016/j.compbiolchem.2021.107563.

[8]Khan, A.A., Shaikh, Z.A., Belinskaja, L., Baitenova, L., Vlasova, Y., Gerzelieva, Z., Laghari, A.A., Abro, A.A., Barykin, S.. A Block chain and Metaheuristic-Enabled Distributed Architecture for Smart Agricultural Analysis and Ledger Preservation Solution: A Collaborative Approach. Applied Sciences, vol.12,no.3,pp.1-19,2022.

[9]Mousapour M., M., Ostadi, A., Pourkhodabakhsh, N., Fathollahi-Fard, A.M., Soleimani, F.. Hybrid neural network-based metaheuristics for prediction of financial markets: A case study on global gold market. Journal of Computational Design and Engineering,vol.10,no.3,pp.1110-1125,2023.

[10]Goldberg, D. E.. Genetic Algorithms in Search Optimization and Machine learning. Addison-Wesley,1989,https://api.semanticscholar.org/CorpusID:38613589.

[11]Storn, R., Price, K.. Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, vol.11,no.4,pp.341-359,1997.

[12]Kennedy, J., Eberhart, R.. Particle Swarm Optimization in Proceedings of ICNN'95 - International Conference on Neural Networks,2002.

[13]Gandomi, A. H., Yang, X. S., Alavi, A. H.. Mixed variable structural optimization using Firefly Algorithm. Computers & Structures, vol.89,no.23-24,pp.2325-2336,2011.

[14]Mirjalili, S., Mirjalili, S. Lewis, A. (2014). Grey Wolf Optimizer. Advances in Engineering Software, vol.69, pp.46-61,2014.

[15]Stutzle, M.. Ant Colony Optimization. Bradford Company,2004.

[16]Karaboga, D., Akay, B.. A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation, vol.214,no.1,pp.108-132,2009.

[17] Wenqing X., Donglin Z., Rui L., Yilin Y., Changjun Z., Shi C.. An effective method for global optimization C Improved slime mould algorithm combine multiple strategies. Egyptian Informatics Journal. 25(100442),pp.1-34,2024.

[18]Mirjalili, S., Lewis, A.. The Whale Optimization Algorithm. Advances in engineering software, vol.95,no.5,pp.51-67,2016.

[19]Faramarzi, A., Heidarinejad, M., Mirjalili, S.,Gandomi, A. H.. Marine Predators Algorithm: A Nature-inspired Metaheuristic. Expert Systems with Applications, vol.152,2020, https://doi.org/10.1016/j.eswa.2020.113377.

[20]Mareli, M., Twala, B.. An adaptive Cuckoo search algorithm for optimisation. Applied Computing and Informatics, vol.14,no.2,pp.107-115,2018.

[21]Hashim, F. A., Houssein, E. H., Hussain K., Mabrouk, M. S., Al-Atabany, W.. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. Mathematics and Computers in Simulation, vol.192, pp.84-110,2022.

[22]Hu, G., Jing, W., Wei, G., Abbas M.. Opposition-based learning boosted orca predation algorithm with dimension learning: a case study of multi-degree reduction for NURBS curves. Journal of Computational Design and Engineering, vol.10,no.2, pp.722-757,2023.

[23]Agushaka, J. O., Ezugwu, A. E., Abualigah, Laith.. Gazelle optimization algorithm: a novel nature-inspired metaheuristic optimizer. Neural Computing and Applications, vol.35,no.5,pp. 4099-4131,2023.

[24]Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., Mirjalili, S. M.. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, vol.114, pp.163C191,2017.

[25]Rao, R. V., Savsani, V. J., Vakharia, D. P.. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design, vol.43,no.3,pp.303-315,2011.

[26]Atashpaz,G. E., Lucas, C.. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In 2007 IEEE Congress on Evolutionary Computation,2007.

[27]Moghdani, R., Salimifard, K.. Volleyball Premier League Algorithm. Applied Soft Computing, vol.64,pp.161-185,2017.

[28]Kuo, H. C., Lin, C. H.. Cultural Evolution Algorithm for Global Optimizations and its Applications. Journal of Applied Research & Technology, vol.11,no.4,pp.510-522,2013.

[29]Zhaolu G., Hongjin L., Kangshun L.. Dual subpopulation artificial bee colony algorithm based on individual gradation. Egyptian Informatics Journal. 25(100452),pp.1-12,2024.

[30]Zhang, S., Zhou, Y.. Grey Wolf Optimizer Based on Powell Local Optimization Method for Clustering Analysis. Discrete Dynamics in Nature and Society,pp.481360(1-17),2015,http://dx.doi.org/10.1155/2015/481360.

[31]Alyu, A. B., Salau, A. O., Khan, B., Eneh, J. N.. Hybrid GWO-PSO based optimal placement and sizing of multiple PV-DG units for power loss reduction and voltage profile improvement. Scientific Reports, vol.13,2023, https://doi.org/10.1038/s41598-023-34057-3.

[32]Lu, C., Gao, L., Li, X., Hu, C., Yan, X., Gong, W.. Chaotic-based grey wolf optimizer for numerical and engineering optimization problems. Memetic Computing, vol.12,pp.371C398,2020.

[33]Mohammad, H. Shahraki, N., Taghian, S., Mirjalili, S.. An improved grey wolf optimizer for solving engineering problems. Expert Systems With Applications, vol.166,pp.113917(1-25),2021, https://doi.org/10.1016/j.eswa.2020.113917.

[34]Su, Y., Li, Y., Xuan, S.. Prediction of complex public opinion evolution based on improved multi-objective grey wolf optimizer. Egyptian Informatics Journal vol.24,pp.149C160,2023.

[35]Li, K., Li, S., Huang, Z., Zhang, M., Xu, Z.. Grey Wolf Optimization algorithm based on Cauchy-Gaussian mutation and improved search strategy. Scientific Reports, vol.12,2022, https://doi.org/10.1038/s41598-022-23713-9.

[36]Tripathi, A. K., Sharma, K., Bala, M. A Novel Clustering Method Using Enhanced Grey Wolf Optimizer and Map Reduce. Big Data Research, 2018,https://doi.org/10.1016/j.bdr.2018.05.002.

[37]Paliwal, N., Srivastava, L., Pandit, M.. Application of grey wolf optimization algorithm for load frequency control in multi-source single area power system. Evolutionary Intelligence, vol.15, pp.563-584,2022.

[38]Song, Y., Meng, X., Jiang, J.. Multi-Layer Perception model with Elastic Grey Wolf Optimization to predict student achievement. PLOS ONE2022, https://doi.org/10.1371/journal.pone.0276943.

[39]Cui, J., Liu, T., Zhu, M., Xu, Z.. Improved team learning-based grey wolf optimizer for optimization tasks and engineering problems. The Journal of Supercomputing,vol.79,pp.10864-10914,2023.

[40]Gupta, S., Deep, K., Moayedi, H., Foong, L. K., Assad, A.. Sine cosine grey wolf optimizer to solve engineering design problems. Engineering with Computers, vol.37,pp.3123-3149,2021.

[41]Pan, C., Si, Z., Du, X., Lv. Y. (2021). A four-step decision-making grey wolf optimization algorithm. Soft Computing, vol.25,pp.14375-14391,2021.

[42]Wang, B., Liu, L., Li, Y., Khishe, M.. Robust Grey Wolf Optimizer for Multimodal Optimizations: A Cross-Dimensional Coordination Approach. Journal of Scientific Computing vol.92,2022, https://doi.org/10.1007/s10915-022-01955-z.

[43]Shial, G., Sahoo, S., and Panigrahi, S.. An Enhanced GWO Algorithm with Improved Explorative Search Capability for Global Optimization and Data Clustering. Applied Artificial Intelligence, vol.37,no.1,2023,https://doi.org/ 10.1080/08839514.2023.2166232.

[44]Yang, Q., Liu, J., Wu, Z., He, S.. A fusion algorithm based on whale and grey wolf optimization algorithm for solving real-world optimization problems. Applied Soft Computing, vol.146,2023,https://doi.org/10.1016/j.asoc.2023.110701

[45] Kai Z.,Chuanhe T.,Yi Z.,Junyuan Y.,Zhilong Z.,Yanqiang W.. Research on Solving Flexible Job Shop Scheduling Problem Based on Improved GWO Algorithm SS-GWO. Neural Processing Letters. p-p.56:26,2024.

[46] Oluwatayomi R. A., Afi K. F., Opeoluwa S. O., Ephraim B. A., Abdelazim G. Hussien, S. K.. Chaotic opposition learning with mirror reflection and worst individual disturbance grey wolf optimizer for continuous global numerical optimization. Scientific Reports. 14:4660,pp.1-41,2024.

[47]Abed-alguni, B.H., Paul, D.. Island-based cuckoo search with elite opposition-based learning and multiple mutation methods for solving optimization problems, Soft Computing. vol.26,no.7,pp.3293-3312,2022.