



ARTICLE

# VGWO: Variant Grey Wolf Optimizer with High Accuracy and Low Time Complexity

Junqiang Jiang<sup>1,2</sup>, Zhifang Sun<sup>1</sup>, Xiong Jiang<sup>1</sup>, Shengjie Jin<sup>1</sup>, Yinli Jiang<sup>3</sup> and Bo Fan<sup>1,\*</sup>

<sup>1</sup>School of Information Science and Engineering, Hunan Institute of Science and Technology, Yueyang, 414006, China

<sup>2</sup>Department of Computer and Information Science, Linköping University, Linköping, 58183, Sweden

<sup>3</sup>Department of Safety and Quality, Hunan Construction Investment Group, Changsha, 410004, China

\*Corresponding Author: Bo Fan. Email: bofan@hnist.edu.cn

Received: 13 May 2023 Accepted: 07 September 2023 Published: 29 November 2023

## ABSTRACT

The grey wolf optimizer (GWO) is a swarm-based intelligence optimization algorithm by simulating the steps of searching, encircling, and attacking prey in the process of wolf hunting. Along with its advantages of simple principle and few parameters setting, GWO bears drawbacks such as low solution accuracy and slow convergence speed. A few recent advanced GWOs are proposed to try to overcome these disadvantages. However, they are either difficult to apply to large-scale problems due to high time complexity or easily lead to early convergence. To solve the abovementioned issues, a high-accuracy variable grey wolf optimizer (VGWO) with low time complexity is proposed in this study. VGWO first uses the symmetrical wolf strategy to generate an initial population of individuals to lay the foundation for the global seek of the algorithm, and then inspired by the simulated annealing algorithm and the differential evolution algorithm, a mutation operation for generating a new mutant individual is performed on three wolves which are randomly selected in the current wolf individuals while after each iteration. A vectorized Manhattan distance calculation method is specifically designed to evaluate the probability of selecting the mutant individual based on its status in the current wolf population for the purpose of dynamically balancing global search and fast convergence capability of VGWO. A series of experiments are conducted on 19 benchmark functions from CEC2014 and CEC2020 and three real-world engineering cases. For 19 benchmark functions, VGWO's optimization results place first in 80% of comparisons to the state-of-art GWOs and the CEC2020 competition winner. A further evaluation based on the Friedman test, VGWO also outperforms all other algorithms statistically in terms of robustness with a better average ranking value.

## KEYWORDS

Intelligence optimization algorithm; grey wolf optimizer (GWO); manhattan distance; symmetric coordinates

## 1 Introduction

This section first introduces why we conduct this research through background and motivation in [Section 1.1](#). Then we present our contribution in [Section 1.2](#).



### 1.1 Background and Motivation

With the development of society, human beings have increasingly higher requirements for the accuracy of engineering design, structural optimization, task scheduling, and other practical engineering applications and scientific research. In the process of conducting the above research, many excellent algorithms have been generated in recent years. For example, the grey wolf optimizer (GWO) [1], the Equilibrium optimizer [2], the Artificial hummingbird algorithm [3], the Multi-trial vector-based monkey king evolution algorithm [4], and so on. They have been shown to outperform many intelligent optimization algorithms and are applied to different engineering optimization problems. However, as the research progressed, many limitations of intelligent optimization algorithms were identified. For example, Zaman et al. used the powerful global exploration capability of the backtracking search optimization algorithm to enhance the convergence accuracy of particle swarm optimization (PSO) [5]. Shen et al. optimized the convergence speed of the whale optimization algorithm by dividing the fitness into three subpopulations [6].

The grey wolf optimizer (GWO) [1] is a swarm-based intelligence optimization algorithm proposed by Mirjalili et al. in 2014. GWO solves the optimization problem by simulating the steps of searching, encircling, and attacking prey in the process of wolf hunting. The first two stages (i.e., searching and encircling) make a global search, and the last stage (i.e., attacking) takes a local search. The social hierarchy of wolves is classified into four types (i.e.,  $\alpha$ ,  $\beta$ ,  $\delta$ , and  $\omega$ ) according to their fitness, where  $\alpha$  wolf has the highest social rank, and  $\omega$  wolf has the lowest social rank. Compared with other optimization algorithms, GWO can adaptively adjust the convergence factor and information feedback to achieve a balance between local optimization and global search, so it has good performance in terms of problem-solving. GWO has been successfully applied to the fields of job shop scheduling [7], parameter optimization [8], and image detection [9] due to its advantages of a simple principle and a few parameters that need adjustment.

Despite the merits mentioned above, concerning the solution accuracy and the searching ability, the original GWO has a potential space to improve. In addition, there is a paradox in GWO: the social rank values of  $\alpha$ ,  $\beta$ , and  $\delta$  are different in accordance with the design of the social hierarchy, but these values are the same in practice. In response to the above questions, Some advanced GWOs have been proposed recently, including variable weight GWO (VWGWO) [10], PSO combined with GWO (PSOGWO) [11], and covariance matrix adaptation with GWO (CMAGWO) [12]. VWGWO strengthens the leadership of  $\alpha$ . PSOGWO accelerates the convergence of populations at later stages. CMAGWO can enhance a local search for wolves. Although these algorithms make some improvements to GWO, they still have the following limitations:

- 1) Low accuracy solution. During the initialization population phase, GWO, VWGWO, and CMAGWO generate wolves randomly; this mechanism cannot guarantee the diversity of the population. Although PSOGWO uses Tent mapping to make the initial population more uniformly distributed, however, due to the fixed mapping period of the Tent mapping, the randomness in the initialization phase still has room to improve. Meanwhile, in our observation, PSOGWO does not remarkably improve the solution quality because of the two fundamental optimizers' common drawbacks of falling into local optimum.
- 2) Unreasonable weight setting. During the searching phase, VWGWO sets the weight of  $\alpha$  close to 1, whereas the weights of  $\beta$  and  $\delta$  are close to 0. This setting will untimely establish the  $\alpha$  wolf's dominance and weaken the leadership of  $\beta$  and  $\delta$ . Thus, VWGWO easily leads to early convergence to  $\alpha$  and falls into a local optimum.

- 3) High time complexity. In CMAGWO, if the GWO solution quality is poor, then the CMA-ES optimization space is extremely limited. In addition, CMA-ES needs to decompose the covariance matrix during iteration (its complexity is as high as  $O(N^3)$ , where  $N$  denotes the number of wolves), and CMAGWO significantly increases the running time compared with GWO.

## 1.2 Our Contributions

To solve the aforementioned problems, this paper proposes a high-accuracy variable GWO (VGWO) while still with low time complexity in this study. The contributions are listed below:

- 1) A mechanism to improve population diversity is developed. Several pairs of wolves with symmetric relationships are added to the population during the wolves' initialization stage. The diversity of the wolves increases and further makes the wolves' distribution more uniform. This scheme is easy to implement and can be applied to problems of different scales.
- 2) A method to prevent wolves from premature convergence is designed. Dynamically adjusting the weights of  $\alpha$ ,  $\beta$ , and  $\delta$  improves the convergence speed and solution accuracy of wolves.
- 3) A notion to make wolves jump out of the local optimum is presented. Random perturbations are leveraged to delay the gathering of the wolves and strengthen the random search ability of the algorithm while keeping the algorithm in a low time complexity.
- 4) Some evaluations are conducted to illustrate the VGWO's advantages. VGWO, GWO, PSOGWO, VWGWO, and CMAGWO are evaluated by 19 benchmark functions and 3 engineering cases. Compared to the state-of-art GWOs and CEC2020 competition winner, the experimental results illustrate that VGWO has good solution quality and fast convergence speed without increasing the time complexity. A further evaluation based on the Friedman test, VGWO also outperforms all other algorithms statistically in terms of robustness with a better average ranking value.

The remainder of this article is as follows: [Section 2](#) reviews the work related to GWO. [Section 3](#) describes the VGWO model and its derivation process. [Section 4](#) gives the parameter settings of different algorithms in the experiments. [Section 5](#) first conducts the experiments on benchmark functions, followed by the analysis of the experimental results, and further demonstrates the excellence of VGWO by comparing its performance with the CEC2020 winner. [Section 6](#) evaluates VGWO through three different engineering examples. [Section 7](#) summarizes the work of this study and provides suggestions for further research.

## 2 Related Work

Many researchers have made relevant improvements to address the shortcomings of GWO. Work related to GWO focuses on three perspectives, namely, the initialization population of GWO, the search mechanism of GWO, and hybridizing other metaheuristics of GWO.

- 1) *Population Initialization of GWO*. The outcome of the initialization population affects the quality and speed of convergence. However, wolves in the original GWO are generated randomly, so their diversity cannot be guaranteed. Long et al. [13] first introduced the point set theory in GWO, which initializes the population and makes wolves' distribution more uniform to solve this problem. Luo et al. [14] employed complex numbers to code the grey wolf individuals. This method can increase the amount of information in the gene because a complex number has 2D properties. Kohli et al. [15] developed a GWO based on chaotic

sequences and demonstrated that their method has greater stability than GWO. Similarly, Teng et al. [11] suggested a particle swarm optimization combined with GWO (PSOGWO), which increases the randomness of initialized populations by Tent mapping. However, the foregoing research only considers the effects of the starting population and ignores the balance between local and global search capabilities.

- 2) *Searching Mechanism of GWO*. A number of studies about search mechanisms focus on the enhancement of control parameter  $a$  ( $a$  is used to control the proportion of global and local search) or on the calculation method of individual coordinates. Saremi et al. [16] combined GWO with evolutionary population dynamics (EPD), where EPD can remove the poor solution and reposition it around  $\alpha$ ,  $\beta$ , or  $\delta$  to improve GWO's local search ability. Mittal et al. [17] proposed a modified GWO (mGWO) that adjusts the control parameter  $a$  to improve the effectiveness and stability of GWO. Lu et al. [18] increased the topological neighbors of wolves to ensure population diversity by combining GWO with topology. However, the weight of dominant wolves ( $\alpha$ ,  $\beta$ , and  $\delta$ ) contradicts their social hierarchy and affects the GWO's performance. Considering this problem, Gao et al. [10] proposed a variable weight GWO to reduce the probability of GWO falling into the local optimum. However, VWGWO's solution quality does not improve much compared to GWO because it overemphasizes the weight of  $\alpha$ .
- 3) *Hybrid Metaheuristics of GWO*. GWO also has gained popularity in the field of hybrid metaheuristics, with many researchers combining it with differential evolution (DE) or PSO. Yao et al. [19] used the survival of the fittest strategy from DE and presented an improved GWO. Similarly, Zhu et al. [20] proposed a hybrid GWO with DE to improve the GWO's global search ability. Contrary to reference [20], Jitkongchuen [21] proposed a hybrid DE algorithm with GWO (jDE). jDE employs the GWO's coordinate update strategy to improve the crossover operator in DE. However, since PSO introduces the historical optimal solution into the algorithm, thereby obtaining strong convergence [22], Singh et al. [23] observed that the above GWOs do not account for wolves' historical optimal coordinates, so they proposed a hybrid GWO based on PSO (HPSOGWO). HPSOGWO has better exploratory ability than GWO; PSOGWO is developed based on HPSOGWO. Zhao et al. [12] used a two stage search to improve the GWO's local search ability and then designed a covariance matrix adaptation with GWO (CMAGWO).

Meanwhile, GWO plays a very important role in the application of engineering problems. In the field of parameter optimization, Madadi et al. [24] used the GWO algorithm to design a new optimal proportional-integral-derivative (PID) controller, then compared the PID-GWO and PID-PSO controllers; the results showed that the PID-GWO can better improve the dynamic performance of the system. Lal et al. [25] employed GWO algorithm to optimize the fuzzy PID controller and applied it to the automatic control of the interconnected hydrothermal power system. Sweidan et al. [26] leveraged GWO to optimize the parameters of the support vector machine (SVM) and used the optimized SVM to assess water quality; Eswaramoorthy et al. [27] utilized GWO to tune the SVM classifier and used it for a case study of intracranial electroencephalography (iEEG) signal classification middle. Muangkote et al. [28] applied the improved GWO algorithm to the training of q-Gaussian-based radial basis function logic network (RBFLN) neural network and pointed out that the improved GWO algorithm has higher accuracy in solving complex problems through comparative experiments; Mirjalili et al. [29] first used GWO to train multi-layer perceptron (MLP). Compared with other well-known algorithms, GWO-MLP can effectively avoid falling into local optimum. Al-Shaikh et al. [30] used GWO to find strongly connected components in directed graphs in linear

time, addressing the serious time-consuming shortcomings of Tarjan's algorithm and the Forward-Backward algorithm. In the field of renewable energy systems, Sivaranjan et al. [31] made use of GWO to determine the optimal feasible solution to the dynamic economic dispatch problem of combined heat and power generation considering wind farms and tested the performance of the GWO algorithm with a test system containing 11 generating units. The results show that the GWO algorithm is superior in both economy and computing time. Mohammad et al. [32] used GWO to solve the problem of photovoltaic solar cells and greatly reduced the waste of energy by estimating the performance. In addition, Almazroi et al. applied GWO to an implicit authentication method for smartphone users [33]. Zaini et al. predicted the energy consumption of home appliances through GWO [34].

Although the above GWOs have some advantages in different parts, they mainly focus on optimizing the internal shortcomings of wolf individuals. This study continues to analyze GWO. Different from the above algorithm, our work focuses on trying to leverage multiple mechanisms to improve the quality of the solution.

### 3 Design of VGWO Algorithm

This section first introduces VGWO in detail. Then, an outline of three aspects is presented, followed by a detailed description of the algorithmic flow. All variants used in this study and their corresponding definitions, as well as the time complexity analysis of VGWO, are also introduced in this section.

#### 3.1 Enhancing Wolves' Diversity through Symmetrical Wolf (SW)

In GWO, initializing the population is a crucial stage. A healthy population has a direct influence on the speed and outcome of convergence in general. However, the original GWO population initialization was performed at random, which cannot guarantee a good diversity of wolves, thereby decreasing the algorithm's effectiveness. This paper gives a definition named SW to overcome this drawback and then focuses on how SW be used to start populations.

**Definition 1: SW.** *Wolf individual's symmetric coordinates about the center of the search range.*

We briefly describe the above definition through an example. Assuming that a point  $x \in [l, r]$  exists in the 1D space, and the SW point of  $x$  is the symmetrical point of  $x$  about the symmetry center  $o = \frac{l+r}{2}$ . When extending to 3D space, for example, a point  $x = (1, 2, 3)$  is found in Fig. 1, where the search range is  $\{(x, y, z) | x \in [-5, 5], y \in [-5, 5], z \in [-5, 5]\}$ , that is, the green space in Fig. 1. Thus, the symmetry center is  $(0, 0, 0)$ , and the SW point of  $x$  in the search range is  $x' = (-1, -2, -3)$ .

Combined with **Definition 1**, the steps of initializing an individual coordinates solution using SW are as follows: 1) Initialize the coordinates  $\vec{X}_i$  ( $i = 1, 2, \dots, N$ ) of  $N$  grey wolf individuals in the defined domain space as the initialization wolves *ini\_population*; 2) Generate the SW population *sym\_population* of population *ini\_population* in accordance with **Definition 1**; 3) Merge populations *ini\_population* and *sym\_population* to obtain *population*, and then sort *population* by fitness value; 4) Select the first  $N$  individuals as the initial population.

The pseudo-code of the initialized population is shown in Algorithm 1, where  $N$  and  $dim$  indicate the amount and dimension of the population, respectively, and  $[l, r]$  represents the search range of wolves. The initialized individuals obey the beta distribution because the beta distribution can generate as many non-edge individuals as possible, thereby improving the search efficiency of GWO. With the use of SW constraint in Algorithm 1, the diversity of the wolves is enhanced considerably. The search

for SW and the initialization of the population are performed simultaneously, which has minimal effect on the time complexity of the algorithm.

---

**Algorithm 1:** The initialize population algorithm

---

**Input:**  $N$ ,  $dim$ ,  $l$ , and  $r$ .

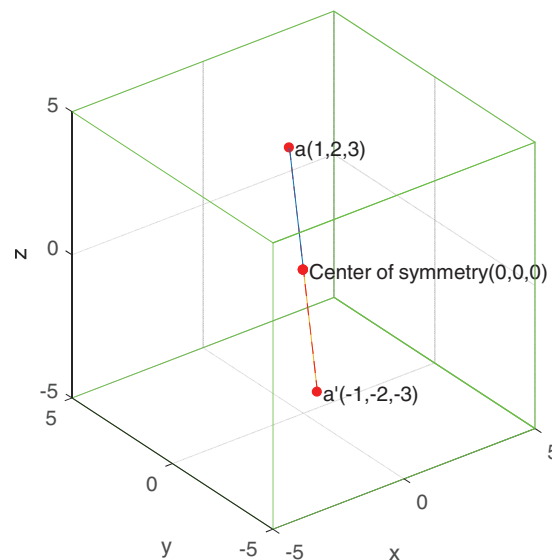
Output:  $population$ .

```

1: procedure SYMMETRIC( $N$ ,  $dim$ ,  $l$ ,  $r$ )
2:   Set arrays  $ini\_pop$  and  $sym\_pop$ .
3:   for ( $i = 0$ ;  $i < N$ ;  $i++$ ) do
4:     for ( $j = 0$ ;  $j < dim$ ;  $j++$ ) do
5:        $ini\_pop[i][j] \leftarrow Beta(l, r)$ .
6:        $sym\_pop[i][j] \leftarrow l + r - ini\_pop[i][j]$ .
7:    $population \leftarrow ini\_pop + sym\_pop$ .
8:   Sort( $population$ ). !according to fitness!
9:   Remove the last  $N$  individuals in  $population$ .
10:  return  $population$ ;

```

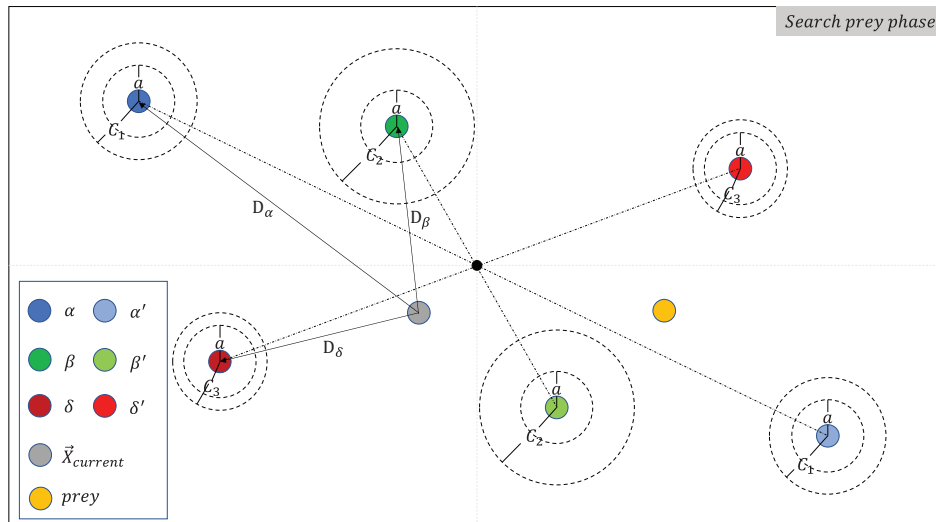
---



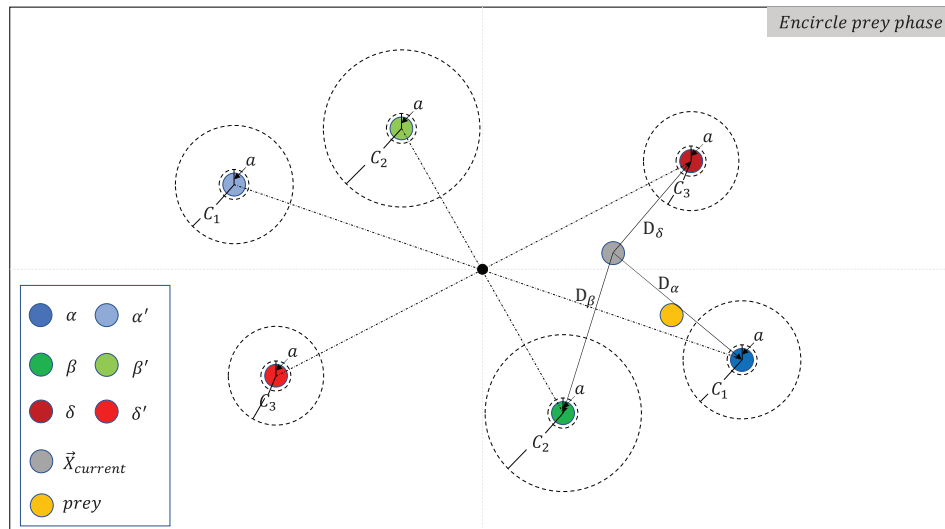
**Figure 1:** SW point in three-dimensional space (1,2,3)

The SW implementation is simple and scalable. Thus, it can be applied after each wolf's coordinate update. Figs. 2 and 3 show the wolves after seeking for SWs in the searching and encircling phases, respectively. The black dot in the figure is assumed to be the search range's center,  $\alpha$ ,  $\beta$ , and  $\delta$  are correspondingly the SWs of  $\alpha$ ,  $\beta$ , and  $\delta$ . The grey point represents an ordinary wolf who will move toward  $\alpha$ ,  $\beta$ , and  $\delta$ , and the yellow point represents the prey. As shown in Fig. 2, SW increases wolf diversity and has the potential to bring wolves closer to their prey. However, the wolves begin to round up the prey, as shown in Fig. 3. If we search SW for the entire population as usual, then the generated SW wolves will be far away from the prey, which is obviously an invalid operation. Thus, the wolves must generate SW wolves adaptively at various stages. In short, VGWO will reduce the SW wolf search when the wolves reach the encirclement phase.





**Figure 2:** Add SW in the phase of searching prey



**Figure 3:** Add SW in the phase of encircling prey

For the SW wolves search in the iterative process, our study only searches for SW wolves for the first *length* of well-fitness individuals in each iteration. This *length* reduces linearly from  $N$  to 0 with the number of iterations. The computation formula is expressed as follows:

$$length = N * \frac{Max\_iter - it}{Max\_iter}, \tag{1}$$

where  $N$  indicates the number of individuals in the population,  $it$  indicates the current iteration, and  $Max\_iter$  indicates the number of iterations. This practice can prevent the time complexity of VGWO from becoming high.

### 3.2 Changing the Weights of $\alpha$ , $\beta$ , and $\delta$

As previously shown, changing the weight of the dominant wolf (i.e.,  $\alpha$ ,  $\beta$ , and  $\delta$ ) can lead to better optimization capability [10], however, the weight of  $\alpha$  should not always be greater than one-third. Although  $\alpha$  is representative of the current optimal solution, it is not constantly closest to the prey [1]. If  $\alpha$  always has the highest weight, then the wolves will inevitably gather around  $\alpha$  prematurely, thereby ignoring the leadership of  $\beta$  and  $\delta$ , and making the search easier to fall into the local optimum. Therefore, this paper proposes a new weight calculation method.

$$\begin{cases} \omega_1 = \frac{2}{3} * \frac{it}{Max\_iter} \\ \omega_2 = \frac{1}{3} \\ \omega_3 = \frac{2}{3} - \omega_1, \end{cases} \quad (2)$$

where  $\omega_1$ ,  $\omega_2$ , and  $\omega_3$  indicate the weight of  $\alpha$ ,  $\beta$ , and  $\delta$ , respectively.  $\omega_1$  linearly increases from 0 to 2/3 with the iterations  $it$ .  $\omega_2$  indicates the weight of the individual in the middle position, and this paper simply keeps its weight to 1/3.  $\omega_3$  linearly reduces from 2/3 to 0 with the number of iterations. As shown in Eq. (2), this calculation method aims to weaken the weight of  $\alpha$  in the searching and encircling phases and strengthen the leadership of  $\beta$  and  $\delta$ , thereby improving the global search ability and reducing the probability of wolves falling into the local optimum.

### 3.3 Adding Random Perturbations to Wolves

Random perturbation is a type of contingent fluctuation that can be used to test the stability and usefulness of a model [35]. Simulated annealing (SA) jumps out of local optimum by implementing a survival of the fittest strategy for random perturbations [36] like the greedy strategy proposed in DE [37]. On this basis, SA and DE have powerful random search capabilities. Contrary to GWO, the authors in [38] and [39] pointed out that SA and DE often need to pay the price of slow convergence speed. Therefore, this paper combines the advantages of the above three algorithms to create the VGWO, that is, adding random perturbations to the wolves. The random perturbations are generated as follows:

$$\vec{X}_{variation} = \vec{X}_{r_1} + F * (\vec{X}_{r_2} - \vec{X}_{r_3}), \quad (3)$$

where  $r_1$ ,  $r_2$ , and  $r_3$  are three different random integers in the range of  $[1, N]$ , and  $F$  is the scaling parameter. In the DE,  $F$  affects the diversity of the population and has a remarkable effect on convergence and stability. The appropriate  $F$  is difficult to be determined. As mentioned in [40], the upper limit of  $F = 1.2$  is determined on the basis of experience, and no optimization problem requires  $F$  to be greater than 1.2. Therefore, for wolves, after ending each search, creating the variant individuals with different  $F$  is a reasonable means to balance the local search and global search capabilities of the algorithm. For the VGWO, the value of  $F$  is linearly reduced from 1.2 to 0. This paper changes  $F$  in accordance with a special parameter, which is called temperature changing in SA.

$$F = 1.2 - \frac{1.2 * (T_{max} - T)}{T_{max} - T_{min}} \quad (4)$$

where  $T_{max}$  indicates the initial temperature,  $T$  indicates the temperature under the current iteration, and  $T_{min}$  indicates the temperature under steady state. However, the judgment of whether a random perturbation is added to the wolf pack needs to be based on its fitness value.



A method for calculating the probability of a random perturbation being selected is proposed in the archival multiobjective SA algorithm (AMOSa) [41].

$$Probability = \frac{1}{1 + e^{\Delta dom_{avg} * T}}, \tag{5}$$

where  $\Delta dom_{avg}$  is the average domination of the random perturbation.  $T$  denotes the present temperature, and the above formula guarantees that the *probability* value is between 0 and 1.  $\Delta dom_{avg}$  is determined as the following:

$$\Delta dom_{avg} = \left( \sum_{i=1}^k \Delta dom_{i,b} \right) / k, \tag{6}$$

where  $k$  is the number of solutions, and  $\Delta dom_{i,b}$  is the amount of domination about two solutions (i.e.,  $i$  and  $b$ ), which is determined by

$$\Delta dom_{i,b} = \prod_{j=1, f_j(i) \neq f_j(b)}^O (|f_j(i) - f_j(b)| / R_j) \tag{7}$$

where the  $O$  is the number of objectives. In [41], the new solution is obtained by making a random perturbation to the current state, which is similar to the idea of deriving the variant individual in Eq. (3). However, the calculation of the average dominance in the AMOSa is based on a multiobjective problem, and the problem discussed in this study is a single-objective problem. In other words,  $O$  in Eq. (7) is equal to 1, and Eq. (6) represents the average domination of the new solution  $b$  over all the solutions. This strategy is similar to the vector 1-norm. The traditional vector 1-norm calculation method is as follows:  $\|x\|_1 = \sum_{i=1}^N |x_i|$ , where  $x$  denotes an  $N$ -dimensional vector. The mathematical meaning of the 1-norm is the sum of the elements of a 1D vector, and the geometric meaning is the Manhattan distance from the vector  $x$  to the original point.

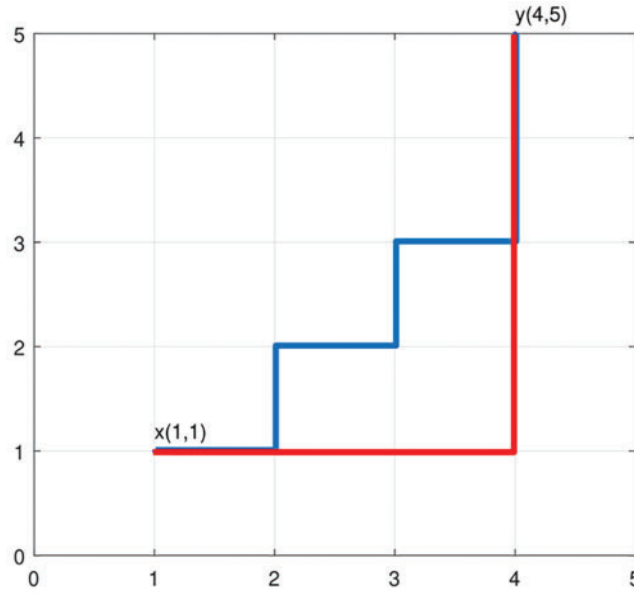
The Manhattan distance is a scalar that reflects the absolute value of the sum of the axial distances between two points. As shown in Fig. 4, the Manhattan distance of point  $x(1,1)$  to point  $y(4,5)$  is 7. However, this study wants to calculate the superiority or inferiority of variant individuals relative to the population as a whole in terms of fitness. Thus, a new method of calculating the Manhattan distance is needed to represent the above problem by the positive or negative of the Manhattan distance. This paper defines this method as the vectorized Manhattan distance to calculate the domination deviation of the variant wolf to the current population. In our study, the fitness value of the population is considered an  $N$ -dimensional vector,  $\vec{Fit} = (Fitness(\vec{X}_1), Fitness(\vec{X}_2), \dots, Fitness(\vec{X}_N))$ . Thus, the corresponding vectorized Manhattan distance calculation is expressed as

$$\vec{V}_{ma} = \overbrace{[Fitness(\vec{X}_{variation}), \dots]}^N - \vec{Fit}, \tag{8}$$

then we calculate the 1-norm  $M$  of non-absolute values of the  $\vec{V}_{ma}$  vector as the following:

$$M = \sum_{i=1}^N V_{ma}[i] \tag{9}$$

where the  $Fitness()$  function is denoted as the fitness function. For example,  $\vec{X}_{variation}$  is better than most individuals in the current wolves when the optimization objective of the fitness function is the minimum, and the value of  $M$  is less than 0. Thus, the vectorized Manhattan distance can be used to determine the status of random perturbations in the current population by the positive or negative of  $M$ . The pseudo-code for vectorized Manhattan distance calculation is presented in Algorithm 2.



**Figure 4:** Manhattan distance from point  $x$  to point  $y$  (both the red and blue lines are Manhattan distances)

---

**Algorithm 2:** The vectorized manhattan distance algorithm

---

**Input:**  $Fitness()$ ,  $population$ ,  $N$ , and  $\vec{X}_{variation}$ .

**Output:**  $M$ ,  $\vec{Fit}$ .

```

1: procedure SYMMETRIC( $Fitness()$ ,  $population$ ,  $N$ ,  $\vec{X}_{variation}$ )
2:   Set arrays  $\vec{Fit}$  and  $M \leftarrow 0$ .
3:    $M += N * Fitness(\vec{X}_{variation})$ .
4:   for ( $i = 0$ ;  $i < N$ ;  $i++$ ) do
5:      $Fit[i] \leftarrow Fitness(population[i])$ .
6:      $M -= Fit[i]$ .
7:   return  $M$ ,  $\vec{Fit}$ ;

```

---

Combined with Eqs. (5) and (6), this paper can derive a new method for calculating the probability of a variation individual being selected is presented as follows:

$$P = \frac{1}{1 + e^{\Delta dom_{avg} * T}} = \frac{1}{1 + e^{(\sum_{i=1}^k \Delta dom_{i,s}) / k * T}} = \frac{1}{1 + e^{\frac{M}{N} * T}} \quad (10)$$

where  $N$  indicates the population number and  $T$  indicates the current temperature.

To better clarify the definition of variables, we list the variables used in this paper and give their corresponding descriptions in Table 1.

### 3.4 VGWO Process

Algorithm 3 demonstrates the VGWO pseudo-code. The initialization section must determine the initial temperature  $T_{max}$ , steady state temperature  $T_{min}$ , and temperature change rate  $k$  based on the actual problem. The initialized population  $ini\_population$  is generated on the basis of the

population size and shape; the initialized individuals follow the beta distribution. The SW population *sym\_population* is generated by using the method described in Section 3.1, *ini\_population* and *sym\_population* are merged to obtain *population*, and *population* is sorted by the fitness value. The top *N* individuals are chosen to form the final initial *population*. The total number of iterations *Max\_iter* is calculated.

**Table 1:** R major variants used in this study

Variants	Definition
$\vec{X}_i$	Grey wolf individuals
$[l, r]$	Search range of wolves
$N$	The number of individuals
$dim$	The dimension of the population
$it$	The current iteration
$Max\_iter$	The amount of iterations
$F$	The DE scaling parameter
$\Delta dom_{avg}$	The average domination of the random perturbation
$O$	The number of objectives
$\vec{V}_{ma}$	The corresponding vectorized Manhattan distance
$M$	The 1-norm of non-absolute values of $\vec{V}_{ma}$
$P$	The probability of a variation individual being selected
$T_{max}$	The initial temperature
$T_{min}$	The steady state temperature
$k$	The temperature change rate
$st^{la}$	The scheduling time of the VGWO reaches
$ast_{algorithm}^{la}$	The average scheduling time of the $la$ -th benchmark
$ARV_i$	The average response of the $i$ -th parameter
$ans_{ij}^{la}$	The VGWO optimized result of the $la$ -th benchmark

The second part is the process of slow cooling. Three wolves with the highest status in the wolves are selected as  $\alpha$ ,  $\beta$ , and  $\delta$ . In accordance with Eq. (1), the number of SW wolves to be searched in each iteration (*length*) is calculated. *sym\_population* is initialized as an empty set, which is used to save the SW of the first *length* wolves. The search process of grey wolves is the same as the original GWO. With the gradual decrease in *length*, the SW of the current grey wolf is searched after determining its new coordinates and saved in *sym\_population* until *length* decreases to 0. Whenever the wolves have finished searching, the variation factor  $F$  is calculated by using Eq. (4). The variation is obtained as a new individual  $X_{variation}$  by using Eq. (3), and the domination  $M$  of  $X_{variation}$  is calculated in accordance with Algorithm 2. The probability of selecting  $\vec{X}_{variation}$  is calculated by using Eq. (10). New offspring *population* and *sym\_population* are combined, and the first  $N$  grey wolves are taken as the new *population*.

### 3.5 Complexity Analysis

Unlike the original GWO, the VGWO uses the quick sort to update the population with a time complexity of  $O(n * \log_2 n)$ . Assume the time complexity of the fitness value calculation is set to  $T$ . The length of the population after the merging of the population and the *sym\_population* is denoted by  $L$ , and the value of  $L$  reduces linearly from  $(2 * N)$  to  $N$  with the number of iterations to facilitate the expression of the subsequent complexity. The basic operations' complexities are as follows. 1) Initializing the wolves:  $O(N * dim + 2 * N * \log_2 2 * N)$ ; 2) Wolf searching: The number of iterations is  $Max\_iter$ . The time complexity of updating the coordinates of wolves is  $O(N * dim)$ . The time complexity of calculating the vectorized Manhattan distance and fitness values is  $O(N * T)$ , while updating the population is  $O(L * \log_2 L)$ . Therefore, the overall complexity of VGWO is  $O(Max\_iter * (N * dim))$ .

The GWO, PSOGWO, and VWGWO all have the same time complexity, which is  $O(Max\_iter * (N * dim))$  [1,10,11]. The time complexity of the CMAGWO is  $O(ite1 * (N * dim)) + O((Max\_iter - ite1) * dim)$  [12], where *ite1* denotes the number of iterations of the first stage in the CMAGWO. The time complexity difference between the five algorithms is small, and subtle differences may be observed. Thus, the effect on the performance of the algorithms is negligible in this study.

---

#### Algorithm 3: The VGWO algorithm

---

**Input:** Adaptation function *Fitness()*, Search scope  $[l, r]$ .

**Output:** *population*[0], *Fit*[0].

1: **procedure** VGWO(*Fitness()*,  $l, r$ )

2:   Set  $T_{max}, T_{min}, N, dim, it \leftarrow 0, k$ , and  $T \leftarrow T_{max}$ .

3:   Initialize the *ini\_population* by using Algorithm 1.

4:    $Max\_iter = \log_k \frac{T_{min}}{T_{max}}$ . /\*Calculate the total amount of iterations according to the initial test temperature and steady state temperature\*/

5:   **while** ( $T > T_{min}$ ) **do**

6:     Select the  $\alpha$  wolf,  $\beta$  wolf, and  $\delta$  wolf according to the *population*.

7:     Calculate *length* by using Eq. (1).

8:     Calculate  $a$ .

9:     Set *sym\_population*  $\leftarrow \emptyset$ .

10:    **for** ( $i = 0; i < N; i++$ ) **do**

11:     Set  $\vec{R}_1$  and  $\vec{R}_2$  to be the value calculated by rand(0, 1).

12:     Calculate  $\vec{A}$  and  $\vec{C}$ .

13:     Calculate  $\vec{D}_\alpha, \vec{D}_\beta$ , and  $\vec{D}_\delta$ .

14:     Calculate  $\vec{X}_1, \vec{X}_2$ , and  $\vec{X}_3$ .

15:     Calculate  $\omega_1, \omega_2$ , and  $\omega_3$  by using Eq. (2).

16:     Set *population*[ $i$ ]  $\leftarrow \omega_1 * \vec{X}_1 + \omega_2 * \vec{X}_2 + \omega_3 * \vec{X}_3$ .

17:     **if** (*length* > 0) **then**

18:       Calculate the SW individual *sym* of *population*[ $i$ ] according to Algorithm 1.

19:       Add *sym* to *sym\_population*.

20:       *length*–.

21:     Set  $r_1, r_2$ , and  $r_3$  to the value calculated by randint (1,  $N$ ).

22:     Calculate  $F$  by using Eq. (4).

23:     Calculate  $\vec{X}_{variation}$  by using Eq. (3).

24:     Calculate the value of  $M$  and *Fit* by using Algorithm 2.

---

(Continued)

**Algorithm 3 (continued)**

```

25: Calculate the probability  $P$  of selecting  $\vec{X}_{variation}$  by using Eq. (10).
26: if ( $rand() < P$ ) then
27:      $population[N - 1] \leftarrow \vec{X}_{variation}$ .
28: Combine  $population$  and  $sym\_population$  to form out  $population$ .
29: Sort( $population$ ). /*according to fitness  $\vec{Fit}$ */
30: Select the first  $N$  individuals after sorting form a new  $population$ .
31:  $it++$ .
32:  $T \leftarrow T * k$ .
33: return  $M, \vec{Fit}$ ;
    
```

**4 Experimental Settings**

With the purpose of assessing the capability of the VGWO, we conduct a series of experiments. The algorithms being compared include GWO, VWGWO, PSOGWO, and CMAGWO. This section outlines the experimental steps and parameter setting. The experimental framework and programs used for comparison were implemented in Python and ran on a computer with a 2.90 GHz Intel i7-10700 CPU and 16 GB of RAM. The source code and experimental results are available at <https://github.com/ZhifangSun/VGWO>.

**4.1 Benchmark Function**

Benchmark function is a variety of standard functions derived from the optimization problems encountered by human beings in real life. This paper selected 19 representative benchmark functions from CEC2014 and CEC2020 to test the performance of the above algorithms, which are well-known complex test functions [42]. They can be divided into three main categories, namely, unimodal (F1, F2, F3, F4, F5, F6, F13, F14), multimodal (F7, F8, F9, F10, F11, F12, F19), and fixed-dimension multimodal (F15, F16, F17, F18). As shown in Table 2, the input  $D$ -dimensional vector  $\vec{X} = (x_1, x_2, \dots, x_D)$  is limited by different definition fields. The maximum value and the minimum value in the definition field are  $r$  and  $l$ , respectively, and the optimum of each function is 0. For each tested algorithm, different random populations are used for 15 repeated tests, and the average solutions and standard deviations of all experiments are recorded.

**Table 2:** Benchmark functions and their definition domain

Label	Dim	Expressions	[l,r]
F1	30	$y = \sum_{i=1}^D ix_i^4 + rand [0, 1)$	[-100, 100]
F2	30	$y = \sum_{i=1}^D  x_i  + \prod_{i=1}^D  x_i $	[-100, 100]
F3	30	$y = \sum_{i=1}^D (\sum_{j=1}^i x_j)^2$	[-100, 100]
F4	30	$y = \max_{1 \leq i \leq D}  x_i $	[-100, 100]
F5	30	$y = (\sum_{i=1}^D x_i^2)^2$	[-100, 100]
F6	30	$y = \sum_{i=1}^D  x_i $	[-100, 100]
F7	30	$y = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^D x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^D \cos(2\pi x_i)} + 20 + e$	[-32, 32]
F8	30	$y = \sum_{i=1}^D (x_i^2/4000) - \prod_{i=1}^D \cos(x_i/\sqrt{i}) + 1$	[-100, 100]

(Continued)

**Table 2 (continued)**

Label	Dim	Expressions	[l,r]
F9	30	$y = 1 - \cos \left( 2\pi \sqrt{\sum_{i=1}^D x_i^2} \right) + 0.1 \sqrt{\sum_{i=1}^D x_i^2}$	[-100, 100]
F10	30	$y = 0.1 \{ \sin^2 (3\pi x_1) + \sum_{i=1}^D (x_i - 1)^2 [1 + \sin^2 (3\pi x_i + 1)] + (x_D - 1)^2 [1 + \sin^2 (2\pi x_D)] \} + \sum_{i=1}^D u(x_i, 5, 100, 4)$	[-50, 50]
F11	30	$y = \sum_{i=1}^D [x_i^2 - 10 \cos (2\pi x_i) + 10]$	[-5.12, 5.12]
F12	30	$y = \frac{\pi}{D} \left\{ 10 \sin^2 \left[ \pi \left( 1 + \frac{x_1 + 1}{4} \right) \right] + \sum_{i=1}^{D-1} \left( \frac{x_i + 1}{4} \right)^2 [1 + 10 \sin^2 \left[ \pi \left( 1 + \frac{x_{i+1} + 1}{4} \right) \right]] + \left( \frac{x_D + 1}{4} \right)^2 \right\} + \sum_{i=1}^D u(x_i, 10, 100, 4)$	[-50, 50]
F13	30	$y = \sum_{i=1}^D  x_i + 0.5 ^2$	[-100, 100]
F14	30	$y = \sum_{i=1}^{D-1} [100 (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30, 30]
F15	6	$y = \sum_{i=1}^4 c_i \exp \left( - \sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right) + 3.322$	[0, 1]
F16	4	$y = \sum_{i=1}^{10} [(X - a_i) (X - a_i)^T + c_i]^{-1} + 10.5363$	[0, 10]
F17	4	$y = \sum_{i=1}^5 [(X - a_i) (X - a_i)^T + c_i]^{-1} + 10.1532$	[0, 10]
F18	2	$y = x_1^2 + 2x_2^2 - 0.3 \cos (3\pi x_1 + 4\pi x_2) + 0.3$	[-100, 100]
F19	30	$y = \{ [\sum_{i=1}^D \sin^2(x_i)] - \exp(-\sum_{i=1}^D x_i^2) \} * \exp(-\sum_{i=1}^D \sin^2 \sqrt{ x_i }) + 1$	[-5, 10]

### 4.2 Parameter Setting of VGWO

Orthogonal experimental design is a multi-factor, multi-level experimental design method [43], which is fast, economical, and efficient, and can obtain satisfactory results with a fewer number of experiments. Therefore, the orthogonal test method is often used to compare the nature of parameters and to determine the best combination between different parameters. The parameters that need to be set in advance for this experiment are the rate of temperature change  $k$ , the starting temperature  $T_{max}$ , and the steady state temperature  $T_{min}$ . The three parameters control the global number of iterations  $Max\_iter$ , the variation factor  $F$ , and the admission probability  $P$  of the perturbed individuals. Therefore, different combinations and different levels of the 3 parameters are shown in Table 3.

**Table 3:** Combinations of different values for parameters

Factor/Parameter	Level/Value				
	1	2	3	4	5
① Temperature change rate: $k$	0.99	0.96	0.93	0.90	0.87
② Initial temperature: $T_{max}$	2.5	2	1.5	1	0.5
③ Steady state temperature: $T_{min}$	0.1	0.01	0.001	1e-4	1e-5

Table 3 lists factors/parameters and their corresponding levels/values. The number of orthogonal groups of  $L_{25}(5^3)$  was chosen for this experiment. For each set of orthogonal experiments, five tests are



performed, and the test termination condition is when the scheduling time of the VGWO reaches  $st^{la} = \max \{ast_{GWO}^{la}, ast_{VWGWO}^{la}, ast_{PSOGWO}^{la}, ast_{CMAGWO}^{la}\}$ , where  $ast_{algorithm}^{la}$  denotes the average scheduling time of the  $la$ -th benchmark function after 300 iterations through  $algorithm$ , where  $algorithm \in \{GWO, VWGWO, PSOGWO, CMAGWO\}$ , and  $la \in \mathbf{Label}$ , for the benchmark functions in Table 2. The results of the orthogonal experiments are shown in the GitHub repository mentioned in Section 4.

The performance of the GWOs is linked to the parameter values when faced with different problems, and a tuned combination of parameter values will yield better results. However, the experiments use the same combination of parameters to optimize different benchmark functions for highlighting the performance of VGWO. Thus, the average response value (ARV) of the  $i$ -th parameter combination is calculated by using the following equation:

$$ARV_i = \frac{1}{5|\mathbf{Label}|} \sum_{j=1}^5 \sum_{la \in \mathbf{Label}} \{ans_{ij}^{la} - \min_{i=1, \dots, 25; j=1, \dots, 5} \{ans^{la}\}\} \tag{11}$$

where  $ans_{ij}^{la}$  represents the VGWO optimized result of the  $la$ -th benchmark function of the  $j$ -th when selecting the  $i$ -th set of parameter combinations, and the value of ARV can reflect the performance of each of the 25 sets of parameter combinations.

The ARVs obtained from the orthogonal experiments are shown in Table 4. Table 5 shows the mean value of ARV and the extreme difference in the mean value for each factor at different levels. A smaller value of ARV indicates that this parameter has less difference from the global optimum at the current level, and a larger value of ARV indicates that this parameter has less optimization ability at the current level. The larger the extreme difference in the mean value of ARV indicates that this parameter has more influence on the performance of the algorithm. Therefore, the conclusions based on Tables 4 and 5 are summarized as follows:

1. In accordance with the value of range, the effect of the three parameters on the performance of the algorithm from low to high is  $T_{max}, T_{min}, k$ .
2. Based on each factor's mean ARV value of different levels, the best combination of parameters for the VGWO is suggested to be  $T_{max} = 1, T_{min} = 1e-4$ , and  $k = 0.90$ .
3. The extreme differences of all three parameters are small, less than 0.25, which shows that the algorithm is not sensitive to the values of the parameters, and has robust and stable performance within a wide range of parameter settings.

**Table 4:** Combinations of different values for parameters

Run	Factor			ARV	Run	Factor			ARV
	①	②	③			①	②	③	
1	1	1	1	0.6865	14	3	4	5	0.6417
2	1	2	3	0.6818	15	3	5	2	0.3672
3	1	3	5	0.6163	16	4	1	3	0.4291
4	1	4	2	0.7509	17	4	2	5	0.4875
5	1	5	4	0.7453	18	4	3	2	0.3590
6	2	1	5	0.7010	19	4	4	4	0.3695
7	2	2	2	0.5198	20	4	5	1	0.7188
8	2	3	4	0.5382	21	5	1	2	0.5085

(Continued)

**Table 4 (continued)**

Run	Factor			ARV	Run	Factor			ARV
	①	②	③			①	②	③	
9	2	4	1	0.4124	22	5	2	4	0.3297
10	2	5	3	0.6072	23	5	3	1	0.8427
11	3	1	4	0.4111	24	5	4	3	0.3197
12	3	2	1	0.5423	25	5	5	5	0.4348
13	3	3	3	0.4836					

**Table 5:** Combinations of different values for parameters

Level	$k$	$T_{max}$	$T_{min}$
1	0.69616	0.54724	0.64054
2	0.55572	0.51222	0.50108
3	0.48918	0.56796	0.50428
4	0.47278	0.49884	0.47876
5	0.48708	0.57466	0.57626
Range	0.22338	0.07582	0.16178
Rank	1	3	2

### 4.3 Other Parameter Settings

The parameters of the VGWO are obtained in Section 4.2 (i.e.,  $T_{max} = 1$ ,  $T_{min} = 1e-4$ , and  $k = 0.90$ ), and the parameters of the remaining algorithms (GWO [1], PSOGWO [11], VWGWO [10], CMAGWO [12]) are set in the same way as the parameters of the respective references. The number of iterations of all algorithms is 90, and the number of wolves is 50.

## 5 Experimental Result

This section presents and discusses the results of the proposed VGWO compared with the state-of-art GWOs. Mainly comparing their performance through optimization results and convergence curves.

### 5.1 Result Analysis

The optimal solutions of the above 10 functions and the algorithms that produce this result are recorded in Table 6, where the optimal average solution of each function is marked in boldface. Given that the generation of the wolves is random and has certain instability in the iterative process, the results that are different from the ideal state exist. However, they are determined by the excellence of the algorithm itself in most cases.

**Table 6:** Results of all algorithms for the benchmark (mean and standard deviation of 15 runs)

Label	VGWO		GWO		CMAGWO		PSOGWO		VWGW0		IMODE	
	Mean	stdev	Mean	stdev	Mean	stdev	Mean	stdev	Mean	stdev	Mean	stdev
F1	<b>0.19e-01</b>	0.17e-01	0.53e+00	0.30e+00	2.28e+00	1.38e+00	2.89e+07	1.71e+07	0.64e+00	0.31e+00	3.25e+09	5.63e+08
F2	<b>1.00e-32</b>	9.34e-33	0.78e-01	0.30e-01	2.11e+00	0.63e+00	2.65e+07	7.03e+07	0.63e-01	0.29e-01	6.89e+37	4.77e+37
F3	<b>7.17e-57</b>	1.53e-56	3.13e+01	3.46e+01	2.83e+03	1.49e+03	2.01e+04	5.06e+03	2.61e+01	1.66e+01	8.44e+04	6.91e+03
F4	<b>4.42e-30</b>	7.25e-30	0.37e+00	0.16e+00	2.44e+00	1.09e+00	5.39e+01	7.74e+00	0.64e+00	0.32e+00	8.86e+01	1.95e+00
F5	<b>6.86e-126</b>	1.94e-125	4.18e-07	5.69e-07	0.34e-01	0.41e-01	7.34e+06	6.89e+06	9.83e-07	8.75e-07	3.13e+09	5.65e+08
F6	<b>1.03e-32</b>	4.82e-33	0.40e-01	0.10e-01	0.75e+00	0.22e+00	8.37e+01	4.62e+01	0.36e-01	0.11e-01	1.07e+03	3.79e+01
F7	<b>6.12e-15</b>	2.94e-15	0.55e-02	0.13e-02	0.11e+00	0.48e-01	9.51e+00	1.74e+00	0.58e-02	0.20e-02	2.03e+01	0.22e+00
F8	<b>0.00e+00</b>	0.00e+00	0.22e-01	0.25e-01	0.12e+00	0.27e+00	1.44e+00	0.27e+00	0.18e-01	0.26e-01	1.57e+01	1.00e+00
F9	<b>6.30e-32</b>	4.48e-32	0.46e+00	0.51e-01	0.66e+00	0.16e+00	6.57e+00	1.68e+00	0.54e+00	0.64e-01	2.38e+01	0.94e+00
F10	3.02e+00	0.22e+00	<b>1.95e+00</b>	0.54e+00	2.36e+00	0.66e+00	3.47e+07	3.41e+07	3.89e+00	0.89e+00	8.27e+08	1.39e+08
F11	<b>0.00e+00</b>	0.00e+00	1.55e+01	4.78e+00	7.59e+01	3.07e+01	2.58e+02	2.65e+01	3.49e+01	1.01e+01	4.04e+02	1.95e+01
F12	0.33e+00	0.20e+00	<b>0.11e+00</b>	0.12e+00	0.16e+00	0.62e-01	7.85e+06	6.99e+06	0.76e+00	0.54e+00	3.94e+08	6.61e+07
F13	3.54e+00	0.44e+00	<b>1.67e+00</b>	0.41e+00	1.97e+00	0.37e+00	1.76e+03	1.24e+03	3.04e+00	0.68e+00	5.86e+04	5.69e+03
F14	<b>2.84e+01</b>	0.43e+00	2.84e+01	0.64e+00	4.09e+01	7.73e+00	3.51e+06	3.52e+06	2.88e+01	0.42e+00	1.84e+08	2.36e+07
F15	<b>0.02e+00</b>	0.01e+00	0.02e+00	0.01e+00	0.18e+00	0.11e+00	0.25e+00	0.05e+00	0.05e+00	0.01e+00	0.61e+00	0.31e+00
F16	0.91e+00	1.91e+00	<b>0.63e-01</b>	0.26e-01	6.66e+00	2.74e+00	7.21e+00	1.48e+00	4.04e+00	1.35e+00	8.06e+00	0.47e+00
F17	<b>0.10e+00</b>	0.16e+00	0.60e+00	1.92e+00	6.65e+00	2.99e+00	7.67e+00	0.97e+00	4.09e+00	1.61e+00	8.04e+00	1.02e+00
F18	<b>0.00e+00</b>	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.23e-01	0.36e-01
F19	<b>1.48e-16</b>	1.30e-16	3.13e-14	4.97e-14	1.56e-10	9.85e-11	2.82e-10	1.28e-10	7.67e-13	1.47e-13	2.51e-08	9.66e-09

The results of the 19 benchmark functions are analyzed as follows:

- 1) For all functions, most of the optimal average solutions compared with various algorithms are found by the VGWO. Because the VGWO enhances the diversity of wolves through SW, it outperforms other algorithms in terms of finding the best solution and even produces the ideal solution 0 for functions F8, F11, and F18. In each iteration, VGWO will arrange more  $\omega$  wolves to search for prey, which greatly increases the probability of obtaining the global optimum.
- 2) The standard deviation of F8, F11, and F18 functions is 0, and the other four GWOs easily fall into local optimum, so it is difficult to achieve this level. VGWO generates random perturbations through differential operators, and calculates the probability of acceptance through its own domination. This method provides a chance for the stability of the wolves to jump out of the local optimum, which reflects the strong robustness of VGWO.
- 3) F10 and F12 are multi-peaked functions, F13 is a single-peaked function, and F16 is a fixed-dimensional multi-peaked function. Although the GWO achieves the lead, it does not differ much from the results obtained by the VGWO, and VGWO has good performance for other functions of the same type.

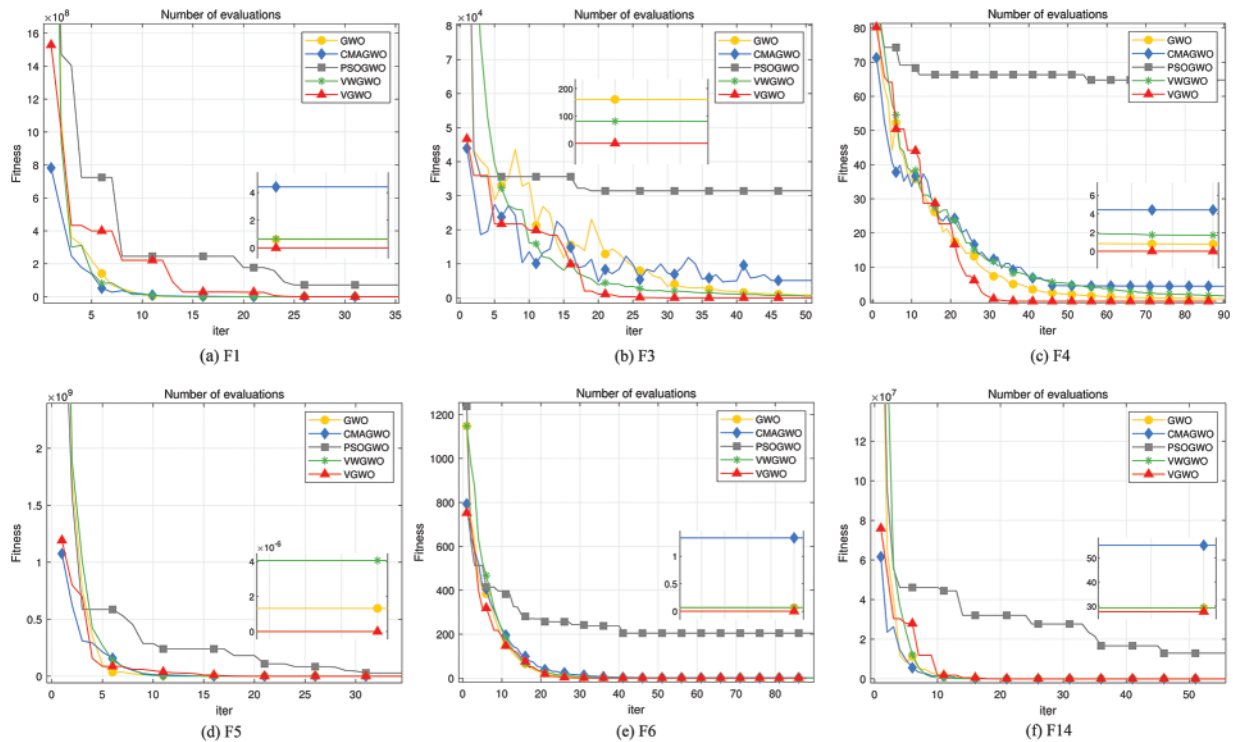
In addition, the Friedman test results shown in [Table 7](#) indicate that VGWO ranks first in terms of average results. It is clear that VGWO is statistically superior to all other algorithms since it has an average ranking value.

**Table 7:** Ranking of all algorithms for all benchmarks obtained by Friedman test

Rank	VGWO	GWO	CMAGWO	PSOGWO	VWGW
F1	1	2	4	5	3
F2	1	3	4	5	2
F3	1	3	4	5	2
F4	1	2	4	5	3
F5	1	2	4	5	3
F6	1	3	4	5	2
F7	1	2	4	5	3
F8	1	3	4	5	2
F9	1	2	4	5	3
F10	3	1	2	5	4
F11	1	2	4	5	3
F12	3	1	2	5	4
F13	4	1	2	5	3
F14	1	2	4	5	3
F15	1	2	4	5	3
F16	2	1	4	5	3
F17	1	2	4	5	3
F18	0.2	0.2	0.2	0.2	0.2
F19	1	2	4	5	3
Average	<b>1.37894737</b>	1.90526316	3.48421053	4.74736842	2.74736842

### 5.2 Convergence Curve Analysis

Figs. 5–7 show most of the fitness value iteration plots. The rest of the functions not shown are due to either excessive disparity or similarities in the convergence curves between different algorithms. GWOs’ excellence cannot be compared visually by the graphs. Considering that the curves of different GWOs overlap in these graphs, some iteration graphs have additional small views, which can be used to visualize the distribution of the overlapping curves at the end of the iteration.



**Figure 5:** Unimodal benchmark functions

The analysis results for the three different types of benchmark functions (unimodal, multimodal, and fixed-dimension multimodal benchmark functions) are as follows:

- 1) For the unimodal benchmark function, the VGWO and CMAGWOs have excellent capabilities in initializing populations due to the beta distribution. Figs. 5b and 5c show that the convergence rate at the beginning is still slow. This finding is due to the improved mechanism proposed in Section 3 enhances the population diversity of the pre-VGWO. However, this scheme does not affect the convergence speed of the VGWO, and the rich population diversity enhances the convergence speed of the VGWO in the later stages. As shown in Figs. 5b and 5c, for the above reasons, the VGWO obtains a solution that exceeds the rest of the algorithms when the number of iterations reaches about the 20th time.
- 2) For the multimodal benchmark function, except for the PSOGWO that will fall into the local optimum prematurely, the rest of the algorithms have a good convergence trend, where the most outstanding performance is still the VGWO. With the continuous addition of random perturbations, and accepting wolves with dominant positions, the VGWO can jump out of the local optimum easily and establish the search area quickly by the rich population diversity.

3) For the fixed-dimension multimodal benchmark function, many algorithms fall into local optimum, and the curves of the GWO, VWGWO, and VGWO show a decreasing trend, as shown in Fig. 7a. In Fig. 7b, only the GWO and the VGWO converge toward the amiable results. In Fig. 7c, only the VGWO converges to the ideal results. The GWO, CMAGWO, PSOGWO, and VWGWO are difficult to determine the approximate position of the prey and present extremely unstable curves. This is due to VGWO's guarantee of population diversity, which reflects the strong robustness of the VGWO.

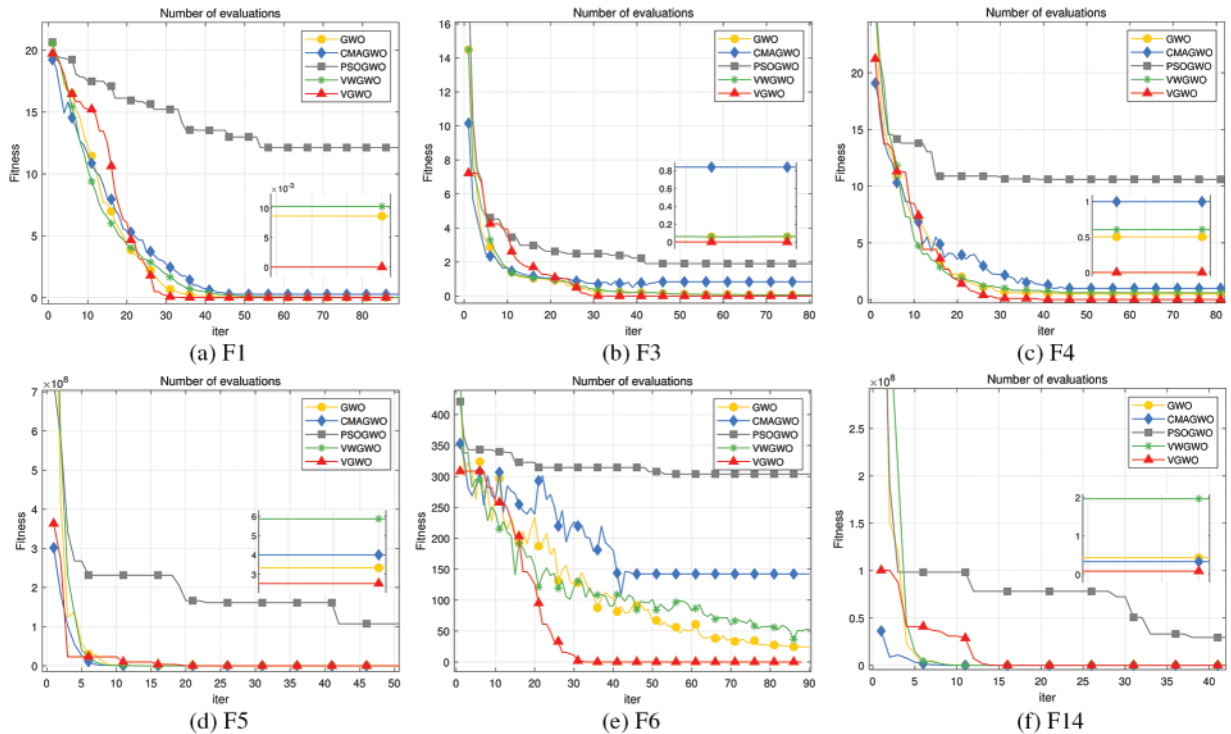


Figure 6: Multimodal benchmark functions

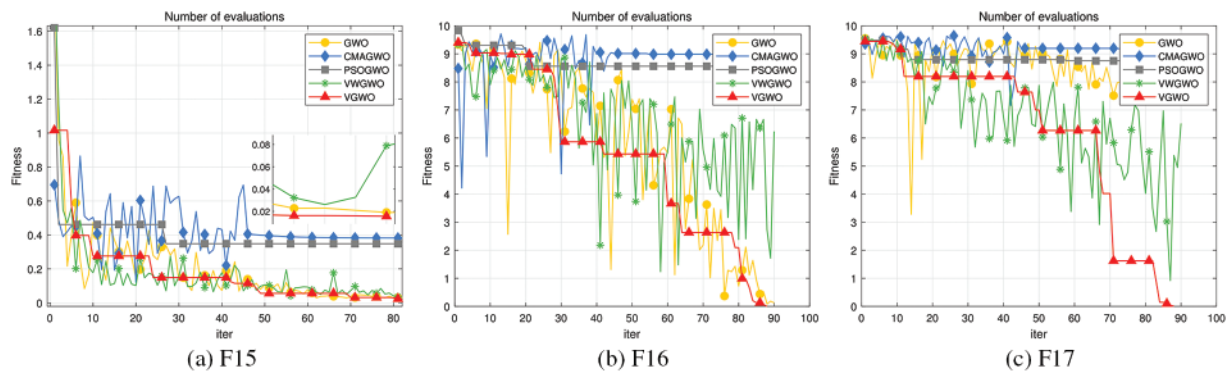


Figure 7: Fixed-dimension multimodal benchmark functions



### 5.3 Comparison with CEC Competition Winner

To judge the performance of VGWO further, this study compares the results obtained by VGWO from the 19 benchmark functions mentioned above with those obtained by the winner of the CEC2020 competition, an improved multi-operator DE (IMODE) algorithm. Its parameters are obtained from relevant articles, and the same random seeds are used for both VGWO and IMODE to ensure a fair comparison.

The mean and standard deviation of the results are recorded according to the rules of the benchmark.

The run was stopped if the number of evolutions was greater than 90. The performance comparison between VGWO and IMODE is shown in [Table 6](#).

By the mean and standard deviation obtained, the proposed VGWO outperforms IMODE for unimodal, multimodal, and fixed-dimension multimodal problems. In [44], IMODE obtains good results by a large number of iterations and running time. However, in this study, the results obtained by IMODE are much worse compared to VGWO. This shows that VGWO is able to obtain better results after fewer iterations compared to IMODE, proving that VGWO has higher accuracy and convergence.

## 6 Application Studies on Engineering Cases

Engineering examples are designed in such a way that the solution usually involves constraints on inequalities or equations. The methods to handle the constraints are special operators, repair algorithms, and penalty functions [45]. The simplest method is used because this study ignores additional algorithms for the VGWO to design good handling of constraints. In simple terms, any solutions that violate the constraints are uniformly penalized by assigning them the maximum fitness value (minimum fitness value in the maximization case). This method is extremely easy to implement and requires no modification to the VGWO.

In the following summary, the VGWO will be used to solve 3 constrained engineering problems and each engineering case will be run 15 times to compare the optimization results with the GWO, CMAGWO, PSOGWO, and VWGWO. The number of iterations of all algorithms is 90, and the number of wolves is 50. The parameter settings for all algorithms are the same as in [Section 4](#).

### 6.1 Cantilever Beam Design Problem

[Fig. 8](#) shows the model of a cantilever beam, which consists of five hollow elements of a square cross section, each corresponding to a variable parameter. From [Fig. 8](#), the right side of element 1 is rigidly supported, and element 5 is subjected to a vertical load [46]. The optimization objective is to minimize the total weight of the beam without violating the vertical displacement constraint. The weight of the beam is calculated by using the following formula:

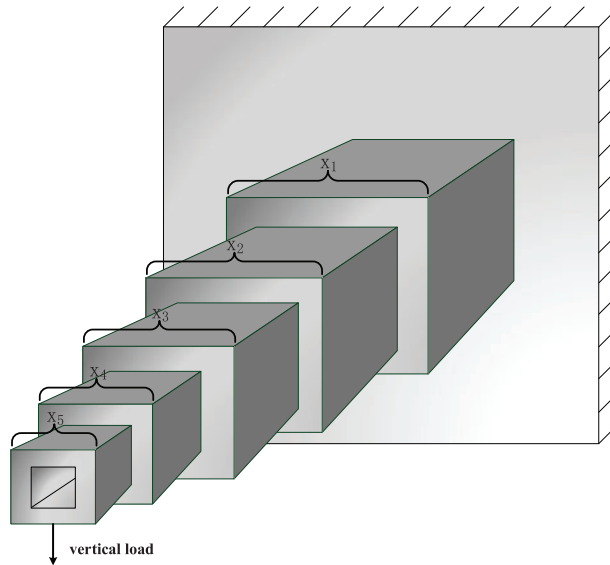
$$f(\vec{x}) = 0.0624 * (x_1 + x_2 + x_3 + x_4 + x_5) \quad (12)$$

The constraint function for the vertical displacement can be expressed as follows:

$$g(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \leq 1 \quad (13)$$

where  $0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100$ . [Table 8](#) shows the best results of the five algorithms optimized for this engineering example. The VGWO achieves the best average result after 15 iterations of the experiment. Although the global optimal solution is obtained by the CMAGWO, its average result is

inferior to that of the VGWO, which reflects the stability of the VGWO. The last column of the table (Max. eval.) represents the maximum number of evaluations required for the algorithm of this row to converge to the average value of the VGWO (bolded values in the table), and N/A represents the inability to reach the average of the VGWO. This finding shows that the VGWO only requires a smaller number of evaluations to converge to a better solution, reflecting the higher accuracy of the results obtained by VGWO.



**Figure 8:** Cantilever beam design problem

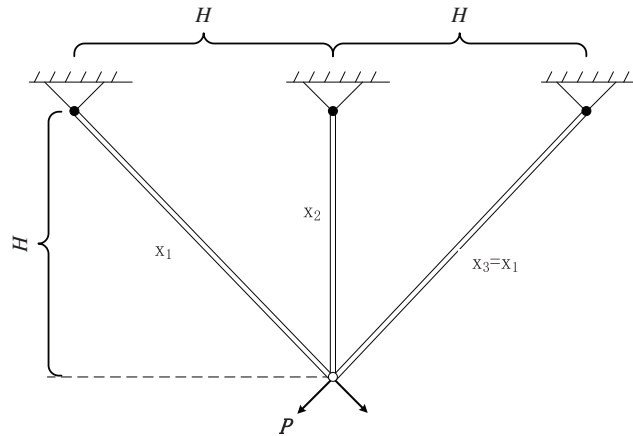
**Table 8:** Comparison results for cantilever design problem

Algorithm	Optimal values for variables					Average of 15 optimizations	Optimum weight	Max. eval.
	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$			
VGWO	6.060636	5.317785	4.434152	3.498035	2.166890	<b>1.3404493</b>	1.3401960	87
GWO	6.071908	5.346120	4.433684	3.501883	2.135901	1.3409640	1.340253	90
CMAGWO	6.023866	5.304281	4.499211	3.508727	2.137856	1.3479516	1.3399740	90
PSOGWO	11.218468	68.452177	92.127824	3.903034	67.669331	1.462091	1.391713	N/A
VWGWO	5.884392	5.143916	4.950015	3.600569	2.111516	1.3621113	1.345435	N/A

## 6.2 Three-Bar Truss Design Problem

Fig. 9 shows the second engineering example, which is the design of a three-bar truss with the optimization objective of minimizing its weight while being constrained by buckling and stresses [47], where trusses  $x_1$  and  $x_3$  are the same. The objective function can be expressed as

$$f(\vec{x}) = (2\sqrt{2}x_1 + x_2) * l \quad (14)$$



**Figure 9:** Three-bar truss design problem

The constraint function can be expressed as

$$\begin{cases} g(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0 \\ g(\vec{x}) = \frac{x_2^2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0 \\ g(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leq 0, \end{cases} \tag{15}$$

where  $0 \leq x_1, x_2 \leq 1, l = 100 \text{ cm}, P = 2 \text{ kN/cm}^2, \sigma = 2 \text{ kN/cm}^2$ . From [Table 9](#), the VGWO achieves the optimal average result, obtains the global optimal solution among the five algorithms, and only uses 79 iterations. The optimal solutions obtained by GWO, PSOGWO, and VWGWO are less than the average of VGWO, which again proves the efficiency of the VGWO in solving engineering instance problems.

**Table 9:** Comparison results of the three-bar truss design problem

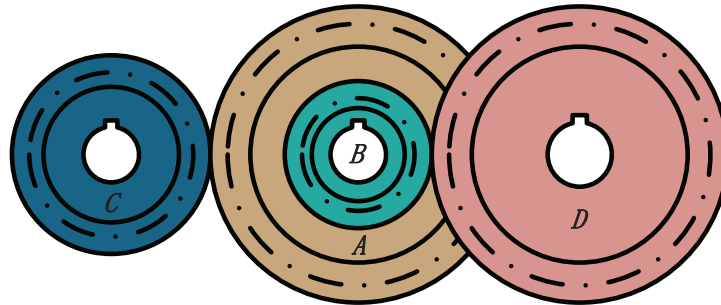
Algorithm	Optimal values for variables		Average of 15 optimizations	Optimum weight	Max. eval.
	$x_1$	$x_2$			
VGWO	0.78833698	0.40923986	<b>263.9313018498927</b>	263.89586755920436	79
GWO	0.79654379	0.4133241	264.1915141452602	263.9365176832555	N/A
CMAGWO	0.78849488	0.40875837	264.1180455966054	263.8993541076323	90
PSOGWO	0.29714538	0.88195163	264.30746833329135	263.9876372189363	N/A
VWGWO	0.62868634	0.3561151	265.67838791337937	264.10334353791336	N/A

### 6.3 Gear Train Design Problem

[Fig. 10](#) shows the third engineering example, which is a discrete case study with four parameters representing four gears (i.e.,  $A, B, C,$  and  $D$ ), and the optimization objective is to minimize the rotation

ratio of the gears and determine the optimal number of teeth for the four gears at this time [48]. The objective function can be expressed as follows:

$$f(\vec{x}) = \left( \frac{1}{6.931} - \frac{x_2 x_3}{x_1 x_4} \right)^2 \quad (16)$$



**Figure 10:** Gear train design problem

Its constraints are relatively simple and only need to be satisfied as follows:  $12 \leq x_1, x_2, x_3, x_4 \leq 60$ . According to [49], when the optimal number of teeth is set to  $x_1 = 43$ ,  $x_2 = 19$ ,  $x_3 = 16$ , and  $x_4 = 49$ , then the gear rotation ratio will obtain the smallest value. As shown in Table 10, VGWO obtains this result by only through 81 iterations, and its average result is the best among the five algorithms. From the above three engineering examples, the VGWO is suitable for most of optimization problems with constraints and can obtain better results after relatively fewer iterations, which reflects the robustness, stability, and strong convergence of the VGWO.

**Table 10:** Comparison results of the gear train design problem

Algorithm	Optimal values for variables				Average of 15 optimizations	Optimum weight	Max. eval.
	$x_1$	$x_2$	$x_3$	$x_4$			
VGWO	43	19	16	49	<b>8.99137768137e-10</b>	2.700857148887e-12	81
GWO	51	26	15	52	8.039399153925e-08	2.307815733313e-11	85
CMAGWO	54	22	17	48	3.692037760945e-06	1.166115767811e-10	90
PSOGWO	57	39	17	51	8.941831357334e-09	2.307815733313e-11	86
VWGWO	40	19	15	52	2.107713739241e-06	1.8273802353e-08	N/A

## 7 Conclusion

In this study, this paper proposes a high-accuracy VGWO while still with low time complexity. SW is used to enhance the diversity of the wolves. Inspired by VWGWO's proposal that three dominant grey wolves have different social statuses in different periods, this study improves the linear variation method of the three dominant wolves' weights during the search procedure. With the purpose of reducing the chance of falling under local optimum, a differential variation factor is introduced to generate new individuals. Enlightened by the AMOSA, the probability of accepting the random perturbation is calculated by vectorizing the Manhattan distance when the grey wolf finishes the current search. The experimental comparison of 19 benchmark functions, the CEC competition winner

(IMODE), and 3 engineering cases with GWO, CMAGWO, PSOGWO, and VWGWO reflects the advantages of VGWO in terms of convergence and accuracy. The experimental results show that our VGWO has better solution quality and robustness than other algorithms under different conditions. For 19 benchmark functions, VGWO's optimization results place first in 80% of comparisons to the state-of-art GWOs and the CEC2020 competition winner. Only VGWO's average ranking value is less than 1.5, according to Friedman's test.

The proposed VGWO is promising to be a good choice for solving problems with uncertain search ranges and high requirements for accuracy of results. However, the proposed improvements mainly focus on increasing the wolves' diversity, which is not necessarily applicable to solve all benchmark functions. Tendency to fall into local optimum is still a weakness of the GWO. VGWO is not always the best solution when encountering a larger range of solution spaces than the current popular application problem, and further efforts are needed to solve this problem. The GWO is still a hot topic for future research on optimization algorithms, and the application of VGWO to suitable engineering optimization problems will be considered in the future. For example, it can be applied to renewable energy systems, to workflow scheduling problems in heterogeneous distributed computing systems, or to integrated process planning and scheduling problems. As more excellent intelligent optimization algorithms are proposed, we will try to shift our research direction to these algorithms in the future, such as Mountain Gazelle Optimizer [50], African Vultures Optimization Algorithm [51], and so on.

**Acknowledgement:** We would like to thank the anonymous reviewers for their valuable comments on improving the paper.

**Funding Statement:** This work is partially funded by the China Scholarship Council.

**Author Contributions:** Study conception and design: Junqiang Jiang, Zhifang Sun; data collection: Xiong Jiang; analysis and interpretation of results: Shengjie Jin, Yinli Jiang; draft manuscript preparation: Junqiang Jiang, Zhifang Sun, Bo Fan. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The authors confirm that the data supporting the findings of this study are available within the article.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

- [1] S. Mirjalili, S. M. Mirjalili and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.
- [2] A. Faramarzi, M. Heidarinejad, B. Stephens and S. Mirjalili, "Equilibrium optimizer: A novel optimization algorithm," *Knowledge-Based Systems*, vol. 191, pp. 105190, 2020.
- [3] W. Zhao, L. Wang and S. Mirjalili, "Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 388, no. 1, pp. 114194, 2022.
- [4] M. H. Nadimi-Shahraki, S. Taghian, H. Zamani, S. Mirjalili and M. A. Elaziz, "MMKE: Multi-trial vector-based monkey king evolution algorithm and its applications for engineering optimization problems," *PLoS One*, vol. 18, no. 1, pp. e0280006, 2023.

- [5] H. R. R. Zaman and F. S. Gharehchopogh, "An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems," *Engineering with Computers*, vol. 38, no. Suppl 4, pp. 2797–2831, 2022.
- [6] Y. Shen, C. Zhang, F. S. Gharehchopogh and S. Mirjalili, "An improved whale optimization algorithm based on multi-population evolution for global optimization and engineering design problems," *Expert Systems with Applications*, vol. 215, pp. 119269, 2023.
- [7] G. Komaki and V. Kayvanfar, "Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time," *Journal of Computational Science*, vol. 8, no. 22, pp. 109–120, 2015.
- [8] X. Song, L. Tang, S. Zhao, X. Zhang, L. Li *et al.*, "Grey wolf optimizer for parameter estimation in surface waves," *Soil Dynamics and Earthquake Engineering*, vol. 75, no. 3, pp. 147–157, 2015.
- [9] H. S. Suresha and S. S. Parthasarathy, "Detection of Alzheimer's disease using grey wolf optimization based clustering algorithm and deep neural network from magnetic resonance images," *Distributed and Parallel Databases*, vol. 40, no. 4, pp. 627–655, 2022.
- [10] Z. M. Gao and J. Zhao, "An improved grey wolf optimization algorithm with variable weights," *Computational Intelligence and Neuroscience*, vol. 2019, pp. 2981282, 2019.
- [11] Z. J. Teng, J. L. Lv and L. W. Guo, "An improved hybrid grey wolf optimization algorithm," *Soft Computing*, vol. 23, no. 15, pp. 6617–6631, 2019.
- [12] Y. T. Zhao, W. G. Li and A. Liu, "Improved grey wolf optimization based on the two-stage search of hybrid CMA-ES," *Soft Computing*, vol. 24, no. 2, pp. 1097–1115, 2020.
- [13] W. Long and T. Wu, "Improved grey wolf optimization algorithm coordinating the ability of exploration and exploitation," *Control and Decision*, vol. 32, no. 10, pp. 1749–1757, 2017.
- [14] Q. Luo, S. Zhang, Z. Li and Y. Zhou, "A novel complex-valued encoding grey wolf optimization algorithm," *Algorithms*, vol. 9, no. 1, pp. 4, 2015.
- [15] M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *Journal of Computational Design and Engineering*, vol. 5, no. 4, pp. 458–472, 2018.
- [16] S. Saremi, S. Z. Mirjalili and S. M. Mirjalili, "Evolutionary population dynamics and grey wolf optimizer," *Neural Computing and Applications*, vol. 26, no. 5, pp. 1257–1263, 2015.
- [17] N. Mittal, U. Singh and B. S. Sohi, "Modified grey wolf optimizer for global engineering optimization," *Applied Computational Intelligence and Soft Computing*, vol. 2016, pp. 7950348, 2016.
- [18] C. Lu, L. Gao and J. Yi, "Grey wolf optimizer with cellular topological structure," *Expert Systems with Applications*, vol. 107, pp. 89–114, 2018.
- [19] P. Yao and H. Wang, "Three-dimensional path planning for UAV based on improved interfered fluid dynamical system and grey wolf optimizer," *Control and Decision*, vol. 31, no. 4, pp. 701–708, 2016.
- [20] A. Zhu, C. Xu, Z. Li, J. Wu and Z. Liu, "Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked soc," *Journal of Systems Engineering and Electronics*, vol. 26, no. 2, pp. 317–328, 2015.
- [21] D. Jitkongchuen, "A hybrid differential evolution with grey wolf optimizer for continuous global optimization," in *2015 7th Int. Conf. on Information Technology and Electrical Engineering (ICITEE)*, Chiang Mai, Thailand, IEEE, pp. 51–54, 2015.
- [22] X. Xia, L. Gui, F. Yu, H. Wu, B. Wei *et al.*, "Triple archives particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 4862–4875, 2020.
- [23] N. Singh and S. Singh, "Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance," *Journal of Applied Mathematics*, vol. 2017, pp. 2030489, 2017.
- [24] A. Madadi and M. M. Mahmood, "Optimal control of DC motor using grey wolf optimizer algorithm," *Technical Journal of Engineering and Applied Sciences*, vol. 4, no. 4, pp. 373–379, 2014.
- [25] D. K. Lal, A. K. Barisal and M. Tripathy, "Grey wolf optimizer algorithm based fuzzy PID controller for AGC of multi-area power system with TCPS," *Procedia Computer Science*, vol. 92, no. 1, pp. 99–105, 2016.
- [26] A. H. Sweidan, N. El-Bendary, A. E. Hassanien, O. M. Hegazy and A. E. Mohamed, "Water quality classification approach based on bio-inspired gray wolf optimization," in *7th Int. Conf. of Soft Computing and Pattern Recognition (SoCPaR)*, Fukuoka, Japan, IEEE, pp. 1–6, 2015.



- [27] S. Eswaramoorthy, N. Sivakumaran and S. Sankaranarayanan, "Grey wolf optimization based parameter selection for support vector machines," *COMPEL-The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 35, no. 5, pp. 1513–1523, 2016.
- [28] N. Muangkote, K. Sunat and S. Chiewchanwattana, "An improved grey wolf optimizer for training q-Gaussian radial basis functional-link nets," in *Int. Computer Science and Engineering Conf. (ICSEC)*, Khon Kaen, Thailand, IEEE, pp. 209–214, 2014.
- [29] S. Mirjalili, "How effective is the Grey Wolf optimizer in training multi-layer perceptrons," *Applied Intelligence*, vol. 43, no. 1, pp. 150–161, 2015.
- [30] A. Al-Shaikh, B. A. Mahafzah and M. Alshraideh, "Metaheuristic approach using grey wolf optimizer for finding strongly connected components in digraphs," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 16, pp. 4439–4452, 2019.
- [31] G. Sivarajan, S. Subramanian, N. Jayakumar and E. B. Elanchezian, "Dynamic economic dispatch for wind-combined heat and power systems using grey wolf optimization," *International Journal of Applied Mathematics and Computer Science*, vol. 2, no. 3, pp. 24–32, 2015.
- [32] M. AlShabi, C. Ghenai, M. Bettayeb, F. F. Ahmad and M. E. H. Assad, "Multi-group grey wolf optimizer (MG-GWO) for estimating photovoltaic solar cell model," *Journal of Thermal Analysis and Calorimetry*, vol. 144, no. 5, pp. 1655–1670, 2021.
- [33] A. A. Almazroi and M. M. Eltoukhy, "Grey wolf-based method for an implicit authentication of smartphone users," *Computers, Materials & Continua*, vol. 75, no. 2, pp. 3729–3741, 2023.
- [34] H. G. Zaini, "Forecasting of appliances house in a low-energy depend on grey wolf optimizer," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 2303–2314, 2022.
- [35] A. V. Skorokhod, F. C. Hoppensteadt and H. D. Salehi, "Random perturbation methods with applications in science and engineering," in *Springer Science & Business Media*, 1st ed., vol. 150. New York, USA: Springer, 2007.
- [36] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [37] R. Storn and K. Price, "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.
- [38] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang *et al.*, "A selfadaptive differential evolution algorithm for scheduling a single batchprocessing machine with arbitrary job sizes and release times," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2019.
- [39] J. Bi, H. Yuan, S. Duanmu, M. Zhou and A. Abusorrah, "Energyoptimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, 2020.
- [40] R. Mallipeddi, P. N. Suganthan, Q. K. Pan and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011.
- [41] S. Bandyopadhyay, S. Saha, U. Maulik and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: Amosa," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269–283, 2008.
- [42] J. J. Liang, B. Y. Qu and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," in *Technical Report 201311*. Singapore: Nanyang Technological University, vol. 635, no. 2, 2013.
- [43] Y. W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.
- [44] K. M. Sallam, S. M. Elsayed, R. K. Chakraborty and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *2020 IEEE Cong. on Evolutionary Computation (CEC)*, Glasgow, UK, IEEE, pp. 1–8, 2020.
- [45] J. Del Ser, E. Osaba, D. Molina, X. S. Yang, S. Salcedo-Sanz *et al.*, "Bioinspired computation: Where we stand and what's next," *Swarm and Evolutionary Computation*, vol. 48, no. 1, pp. 220–250, 2019.

- [46] S. Li, H. Chen, M. Wang, A. A. Heidari and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.
- [47] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan *et al.*, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm and Evolutionary Computation*, vol. 56, pp. 100693, 2020.
- [48] A. H. Gandomi, "Interior search algorithm (ISA): A novel approach for global optimization," *ISA Transactions*, vol. 53, no. 4, pp. 1168–1183, 2014.
- [49] A. H. Gandomi, X. S. Yang and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, 2013.
- [50] B. Abdollahzadeh, F. S. Gharehchopogh, N. Khodadadi and S. Mirjalili, "Mountain gazelle optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems," *Advances in Engineering Software*, vol. 174, no. 3, pp. 103282, 2022.
- [51] B. Abdollahzadeh, F. S. Gharehchopogh and S. Mirjalili, "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems," *Computers & Industrial Engineering*, vol. 158, no. 4, pp. 107408, 2021.