Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 21(3): 3910–3943.
DOI: 10.3934/mbe.2024174
Received: 30 January 2024
Accepted: 18 February 2024
Revised: 14 February 2024
Published: 21 February 2024

Research article

Adaptive dynamic self-learning grey wolf optimization algorithm for solving global optimization problems and engineering problems

Yijie Zhang and Yuhang Cai*

School of Artificial Intelligence and Computer Science, Jiangnan University, WuXi 214122, China

* **Correspondence:** Email: 6223110013@stu.jiangnan.edu.cn.

Abstract: The grey wolf optimization algorithm (GWO) is a new metaheuristic algorithm. The GWO has the advantages of simple structure, few parameters to adjust, and high efficiency, and has been applied in various optimization problems. However, the orginal GWO search process is guided entirely by the best three wolves, resulting in low population diversity, susceptibility to local optima, slow convergence rate, and imbalance in development and exploration. In order to address these shortcomings, this paper proposes an adaptive dynamic self-learning grey wolf optimization algorithm (ASGWO). First, the convergence factor was segmented and nonlinearized to balance the global search and local search of the algorithm and improve the convergence rate. Second, the wolves in the original GWO approach the leader in a straight line, which is too simple and ignores a lot of information on the path. Therefore, a dynamic logarithmic spiral that nonlinearly decreases with the number of iterations was introduced to expand the search range of the algorithm in the early stage and enhance local development in the later stage. Then, the fixed step size in the original GWO can lead to algorithm oscillations and an inability to escape local optima. A dynamic self-learning step size was designed to help the algorithm escape from local optima and prevent oscillations by reasonably learning the current evolution success rate and iteration count. Finally, the original GWO has low population diversity, which makes the algorithm highly susceptible to becoming trapped in local optima. A novel position update strategy was proposed, using the global optimum and randomly generated positions as learning samples, and dynamically controlling the influence of learning samples to increase population diversity and avoid premature convergence of the algorithm. Through comparison with traditional algorithms, such as GWO, PSO, WOA, and the new variant algorithms EOGWO and SOGWO on 23 classical test functions, ASGWO can effectively improve the convergence accuracy and convergence speed, and has a strong ability to escape from local optima. In addition, ASGWO also has good performance in engineering problems (gear train problem, ressure vessel problem, car crashworthiness problem) and feature selection.

Keywords: grey wolf optimization; global optimization; metaheuristics; real engineering problems; self-learning; adaptive dynamics

1. Introduction

The pursuit of the optimal solution among numerous alternatives to either maximize or minimize the objective function, while adhering to constraints, constitutes an optimization problem. Such challenges manifest ubiquitously across various domains including but not limited to signal processing, image processing, production scheduling, task allocation, pattern recognition, automatic control, and machine design. Optimization algorithms primarily harness the tremendous computational prowess of computers to iteratively explore viable solutions to the problem. After a large number of feasible solutions have been obtained, the most appropriate solution is selected to formulate a computational method for solving the problem [1]. Various optimization methods such as the particle swarm algorithm [2], Whale algorithm [3], and Ant-Lion algorithm [4], etc. have been widely used in the above fields and have yielded great economic and social benefits. Given the myriad variables, intricacies, computational overheads, and nonlinearity inherent in optimization challenges, scientists and engineers globally continue their quest for an efficient and versatile optimization methodology.

The grey wolf optimization algorithm (GWO) [5] is a new heuristic swarm intelligence optimization algorithm proposed by Mirjalili et al. in 2014, that finds applications across diverse domains including engineering, medicine, image processing, and biological sciences. The GWO draws inspiration from the social structure and hunting tactics of grey wolves in the wild. As shown in Figure 1, the wolf pack is specifically divided into four ranks. Through the guidance of the three alpha wolves, the grey wolves conduct collective searches, encirclement, and attacks on prey, realizing the search for targets. Owing to its remarkable efficiency and minimal adjustable parameters, the GWO algorithm is characterized by ease of implementation. Consequently, in recent years, it has been applied to many fields, such as workshop scheduling [6], path planning [7], power systems [8], fuzzy control systems [9], image segmentation [10], and so on. However, the GWO algorithm's reliance on the top three wolves for guiding the entire search process often results in rapid convergence towards these wolves. Therefore, GWO suffers from the disadvantages of low population diversity, tendency to fall into local optima, slow convergence in later stages, and imbalance in the exploration and exploitation process. Due to these deficiencies, there is a significant gap between GWO and SMAF1 [11] in the optimal tuning of interval type-2 fuzzy controllers. In light of these drawbacks, scholars have proposed a series of improved solutions. For instance, the method proposed in [12] to delete half of the less fit search agents and relocate them near the three best wolves can improve local search and convergence towards promising regions of the search space. In [13], GWO was applied to optimally tune the parameter vectors of a fuzzy control system. With the rapid development of neural networks, more and more improvements have emerged. In [14], the authors proposed an improved anti-noise adaptive long short-term memory (ANA-LSTM) neural network with high-robustness feature extraction and optimal parameter characterization for accurate Remaining Useful Life (RUL) prediction. In [15], an improved robust multi-time scale singular filtering-Gaussian process regression-long short-term memory (SF-GPR-LSTM) modeling method is proposed for remaining capacity estimation, and these methods have achieved good improvement results. Nevertheless, drawing inspiration from [16], there exists a potential to integrate intelligent optimization algorithms with LSTM. GWO could be employed to compute meaningful and optimal hyperparameters for CNN-LSTM networks, yielding notable performance enhancements. In [17], the author introduced random search agents into the position update equation to increase the algorithm's exploration capability. However, in the exploitation stage, there is no limit on the influence of random search agents, which can weaken the local search capability of the algorithm and affect convergence accuracy. The impact on convergence accuracy is particularly significant in constrained engineering design problems. In [18], the author integrated GWO with PSO to improve convergence accuracy, but the weakness of easily falling into local optima remains. In [19], the author introduced a crossover operator between two random individuals to achieve information sharing among individuals, improving convergence speed and solution quality. When the population falls into local optima, individuals gather together, and the crossover operator loses its effect, lacking the ability to escape local optima. In [20], the author incorporated opposition-based learning into GWO to improve convergence speed, but the complexity of the algorithm also increases. In [21], the author used fuzzy logic to dynamically adjust parameters, change the weights of the three alpha wolves, and highlight the leadership disparities of the grey wolf pack to improve convergence accuracy. The original GWO has slow convergence speed, is prone to falling into local optima, has weak search ability, and has low convergence accuracy. To address these shortcomings, this paper proposes an adaptive dynamic self-learning grey wolf optimization algorithm (ASGWO):

1). ASGWO proposes a piecewise nonlinear factor a to achieve a balance between global search and local development.

2). ASGWO integrates a dynamic logarithmic spiral into the foundational position update equation, gradually diminishing its configuration over successive iterations. This augmentation serves to broaden the algorithm's search domain while concurrently enriching population diversity.

3). ASGWO replaces the static step size of the original position update with an adaptive self-learning step size, dynamically adjusting it according to the learning of evolutionary success rate and iteration count. This adaptation enables the algorithm to optimize step size in alignment with current information, thereby enhancing both convergence speed and the algorithm's capability to circumvent local optima.

4). ASGWO also proposes a new location update strategy, using the global optimal location and randomly generated locations as learning samples, and adding dual adaptive convergence factors to control the influence of the two learning samples.

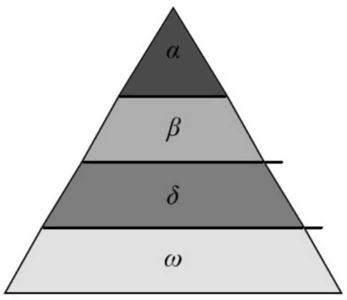

Figure 1. Rank system of the grey wolf.

We endeavor for ASGWO to demonstrate robust performance across 23 test functions and to attain exceptional outcomes when employed in engineering scenarios. Subsequent experimental findings will substantiate this notion.

The rest of the article is structured as follows: Section 2 briefly introduces the mathematical model of the GWO algorithm. Section 3 describes the improvement strategy and implementation steps of ASGWO. Section 4 analyzes the experimental results of the benchmark function. Section 5 shows applications of ASGWO on real engineering problems. Finally, Section 6 presents the conclusions of this article.

2. Grey wolf optimizer

In designing GWO, a mathematical model is constructed for the grey wolf population. The wolf with the best fitness is the α wolf, followed by the β wolf and the δ wolf, and the remaining solutions are the ω wolves. During hunting, ω wolves will approach, surround, and attack prey under the guidance of α wolf, β wolf, and δ wolf. For a d-dimensional optimization problem, the population in GWO consists of multiple grey wolves, each representing a candidate solution. The position vector of the grey wolf represents the feature vector of the corresponding candidate solution. The objective function value of the candidate solution corresponds to the fitness of the grey wolf.

2.1. Encircling prey

The wolf's strategy of surrounding the prey during hunting, to mathematically model it, proposes the following equation:

$$\vec{D} = \left|\vec{C} \times \vec{X_p}(t) - \vec{X}(t)\right| \tag{2.1}$$

$$\vec{X}(t+1) = \vec{X_p}(t) - \vec{A} \times \vec{D} \tag{2.2}$$

$$a(t) = 2 - \frac{2t}{MaxIter} \tag{2.3}$$

$$\vec{A} = 2a \cdot \vec{r_1} - \vec{a} \tag{2.4}$$

$$\vec{C} = 2 \cdot \vec{r_2} \tag{2.5}$$

where the t represents the current iteration number, $MaxIter$ is the total iteration number, $\vec{D}$ is the distance between the wolf and the prey, $\vec{X_p}(t)$ is the position vector of the prey, $\vec{X}(t+1)$ is the position vector of the wolf at iteration t, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{r_1}$ and $\vec{r_2}$ are random vectors in [0,1], the component of a decreases linearly from 2 to 0 during the iteration process, and $\vec{a}$ is a vector composed of scalars a.

2.2. Hunting

In an abstract search space, we do not know the location of the prey. To mathematically simulate the hunting behavior of grey wolves, we assume that α wolves, β wolves, and δ wolves have better knowledge of the potential location of the prey. Therefore, we save the first three best solutions obtained so far and require other search agents to update their positions based on the guidance of the position of the three best search agent. In nature, there are also differences in social hierarchy among the three best wolves. Therefore, this article refers to the fitness weight mentioned in the literature [18] to reflect the differences between the three best wolves, making the algorithm more consistent with the social hierarchy of grey wolves. The fitness of alpha wolf is the best among the three best wolves, so the inertia weight of the alpha wolf is the largest, followed by the delta wolf and the omega wolf. The following formula is proposed in this regard.

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \times \overrightarrow{X_\alpha} - \overrightarrow{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2} \times \overrightarrow{X_\beta} - \overrightarrow{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \times \overrightarrow{X_\delta} - \overrightarrow{X}\right| \tag{2.6}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \times \overrightarrow{D_\alpha}, \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \times \overrightarrow{D_\beta}, \overrightarrow{X_\delta} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \times \overrightarrow{D_\delta} \tag{2.7}$$

$$\overrightarrow{X}(t+1) = \left(W_1 \cdot \overrightarrow{X_1} + W_2 \cdot \overrightarrow{X_2} + W_3 \cdot \overrightarrow{X_3}\right) \tag{2.8}$$

$$W_1 = \frac{Z_\alpha}{Z_\alpha + Z_\beta + Z_\delta}, W_2 = \frac{Z_\beta}{Z_\alpha + Z_\beta + Z_\delta}, W_3 = \frac{Z_\delta}{Z_\alpha + Z_\beta + Z_\delta} \tag{2.9}$$

where the $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$, and $\overrightarrow{X_\delta}$ are the position vectors of the alpha, beta, and delta wolves, and Z_α, Z_β, and Z_δ represent the reciprocal of the fitness of alpha, beta, and delta wolves, respectively.

2.3. Exploration and exploitation in hunting

When the prey stops moving, the grey wolf attacks the prey to complete the hunt. To approach the prey in the mathematical simulation, we reduce the value of a to reduce the fluctuation range of $\overrightarrow{A}$. $\overrightarrow{A}$ is a random value in the range of $[-2a, 2a]$, where a decreases from 2 to 0 with the number of iterations. When the random value of $\overrightarrow{A}$ is between $[-1, 1]$, the next position of the search agent is anywhere between the current position and the prey position. Therefore, when $\left|\overrightarrow{A}\right| < 1$, the wolf group explores the search space. Grey wolves mainly search based on the positions of alpha wolves, beta wolves, and delta wolves. They separate from each other to find prey and converge to attack prey. To mathematically model divergence, we use random values when $|\overrightarrow{A}| > 1$ to force the search agent to deviate from the location of the prey, thus exploring the search space.

The grey wolf completes hunting by repeating the steps of encirclement and hunting as described above. The pseudocode of the original GWO algorithm is shown in Algorithm 1.

Algorithm 1 Grey Wolf Algorithm.

```
1: Initialize the grey wolf population
2: repeat
3:     Calculate parameters a, A, and C
4:     Calculate the fitness of the search agent
5:     Find the three best agents: $\alpha, \beta, \gamma$
6:     Update search agent position through Equation2.8
7: until The conditions for termination are met
Output: optimal solution
```

3. ASGWO

3.1. Segmented nonlinear convergence factor

In the original grey wolf optimization algorithm, the global exploration and local exploitation abilities of the algorithm are determined by the coefficient $\left|\vec{A}\right|$, which is determined by the convergence factor a. The convergence factor a decreases linearly from 2 to 0, imparting upon the algorithm pronounced global exploration capabilities in its nascent phases and robust local exploitation prowess in its subsequent stages. However, the algorithm's convergence exhibits nonlinearity throughout the iterative process. The linear reduction of the convergence factor a cannot well fits the real search situation. In the iterative process of the algorithm, if the linear convergence factor a decreases too fast in the early stage, it may lead to insufficient exploration, and then the algorithm very easily falls into premature convergence in the exploitation process. Conversely, a sluggish reduction of the linear convergence factor a in the later stages can significantly prolong convergence time and diminish convergence efficiency. In the early stages of search, the nonlinear convergence factor a can decrease at a smaller rate to ensure sufficient global exploration; in the later stages of exploitation, the nonlinear convergence factor a decreases faster, thereby improving the convergence speed and enhancing local exploitation. Dividing the iterative process into early-stage exploration and late-stage exploitation can facilitate achieving a harmonious equilibrium between global exploration and local exploitation across a wide spectrum of problems. Therefore, this article advocates for the adoption of a segmented nonlinear convergence factor, with the concrete formula as follows:

$$\begin{Bmatrix} a = 2 - tan^{1.5}\left(\dfrac{2 \cdot t}{MaxIter} \cdot \dfrac{\pi}{4}\right),\ if\ t < \dfrac{MaxIter}{2} \\ a = tan^{1.5}\left(\dfrac{2 \cdot (MaxIter - t)}{MaxIter} \cdot \dfrac{\pi}{4}\right),\ otherwise \end{Bmatrix} \tag{3.1}$$

For x within the interval [0,1], the growth rate of $\tan\left(\frac{\pi}{4}x\right)$ is greater than that of x. Thus, the rate of decrease for $2 - \tan\left(\frac{\pi}{4}x\right)$ is slower compared to $2 - x$. On the other hand, for x within the interval [1,2], the growth rate of $\tan(x)$ is exceeds that of x. Therefore, the descent rate of $\tan(2 - x)$ outpaces that of $2 - x$. To amplify this effect, an exponential function with a base greater than 1 can be applied. As per Eq (3.1), during the initial phase of the iteration process, the nonlinear convergence factor a undergoes a gradual reduction, maintaining a larger value compared to the linear convergence factor, so that the algorithm can explore a broader search space, improve population diversity, and establish a robust groundwork for algorithm exploitation. Subsequently, in the second half of the iteration process, the preliminary comprehensive exploration reduces the possibility of falling into local optimal solutions. Moreover, the nonlinear convergence factor a decreases faster, enabling the algorithm to quickly enter fine local exploitation and improve the convergence speed. Figure 2 illustrates the curve of the nonlinear convergence factor a changing with the number of iterations.

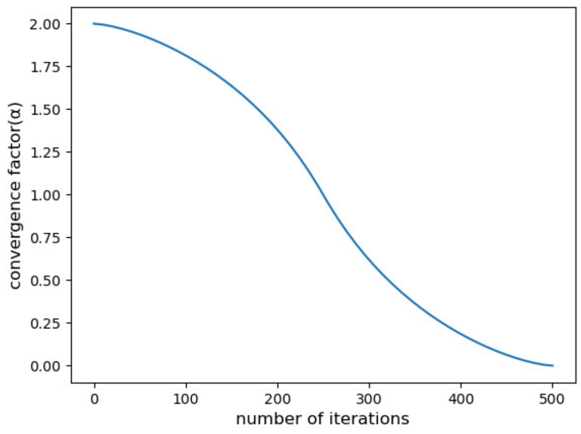

Figure 2. The curve of the nonlinear convergence factor *a* changing with the number of iterations.

3.2. *Dynamic logarithmic spiral*

In the original grey wolf optimization algorithm, the wolf pack moves slowly towards the three best wolves in a straight line to approach the prey, as these top-ranking wolves can acquire a wealth of prey-related data. However, this approach to position updating is overly simplistic, potentially leading to the oversight of crucial information during movement, resulting in a small search range and easy premature convergence. Given the cautious nature of grey wolves, they do not move in a straight line but move slowly in circles to avoid scaring the prey and causing hunting failure. During the ultimate pursuit, straight-line movement enables swift proximity to the prey; however, owing to the prey's evasive maneuvers and the grey wolves' limitations in maintaining straight-line hunting, the final hunt is also curved movement. Therefore, the incorporation of a logarithmic spiral into the position update process offers a more faithful emulation of grey wolf locomotion. Concurrently, as the distance decreases, the curvature of the grey wolf motion also becomes smaller. Therefore, the configuration of the logarithmic spiral is regulated by amalgamating the functions of cos and $\sqrt{\frac{t}{Maxiter}}$ to generate a monotonically decreasing function. With an escalation in the number of iterations, the configuration of the logarithmic spiral is changed to become smaller, aligning more closely with the movement patterns of grey wolves. Therefore, this article proposes a dynamic spiral position update, with the concrete formula as follows:

$$\begin{aligned}
\overrightarrow{X_1} &= \overrightarrow{X_\alpha} - \overrightarrow{A_1} \times \overrightarrow{D_\alpha} \cdot e^{b \cdot l_1} \cdot \cos(2 \cdot \pi \cdot r_3) \\
\overrightarrow{X_2} &= \overrightarrow{X_\beta} - \overrightarrow{A_2} \times \overrightarrow{D_\beta} \cdot e^{b \cdot l_2} \cdot \cos(2 \cdot \pi \cdot r_4) \\
\overrightarrow{X_3} &= \overrightarrow{X_\delta} - \overrightarrow{A_3} \times \overrightarrow{D_\delta} \cdot e^{b \cdot l_3} \cdot \cos(2 \cdot \pi \cdot r_5)
\end{aligned} \tag{3.2}$$

$$b = \cos\left(\sqrt{\frac{t}{MaxIter}} \cdot \pi\right) \tag{3.3}$$

where l_1, l_2, l_3, r_1, r_2, r_3 are random values of [-1,1], respectively.

Utilizing Eq (2.9), wolves can update their location via the logarithmic spiral, traversing regions inaccessible to linear movement, obtaining additional information in the path, expanding the search range of the algorithm, and improving the diversity of the population. The dynamic spiral parameter in Eq (3.1) depends on the number of iterations, resulting in a larger spiral configuration during the initial stages of iteration. This enables the wolves to explore a larger range, further expanding the search range of the algorithm, and circumventing premature convergence with better population diversity; in the later stages of iteration, the spiral shape becomes smaller, enhancing the local development ability of the wolves and improving the convergence speed of the algorithm.

3.3. Dynamic self-learning step size

Equation (2.8) stipulates that the step size of the traditional grey wolf optimization algorithm is fixed. In scenarios where a longer step size is warranted for convergence, a short current step size mandates a gradual approach to the optimal point, thereby impeding convergence speed. Conversely, if a smaller step size is needed for convergence, an excessively large algorithmic step size may result in the search agent's oscillation around the optimal point, perpetually advancing and retreating. Moreover, a single fixed step size cannot make reasonable use of current information, leading the algorithm to fall into locally optimal solutions. Therefore, modulating the algorithm's step size based on the current evolutionary success rate (*ratio*) and iteration count offers a potential solution. When the algorithm needs a longer step size, increasing the step size can improve the convergence speed. When the algorithm needs a shorter step size, reducing the step size can prevent the search agent from oscillating around the optimal point. In the early stages, the step size should be larger to conduct wide-area exploration of the search space, and in the later stages, the step size should be smaller to achieve fine local development of the search space. In addition, when the algorithm falls into local optimal solutions, the ratio will decrease significantly. In this case, a larger step size is needed to help the algorithm escape from locally optimal solutions. Therefore, this article proposes adaptive step adjustment based on evolutionary success rate, with the concrete formula as follows:

$$ratio\,(t+1) = \frac{k\,(t)}{SearchAgents} \tag{3.4}$$

$$\overrightarrow{X}\,(t+1) = \left(W_1 \cdot \overrightarrow{X_1} + W_2 \cdot \overrightarrow{X_2} + W_3 \cdot \overrightarrow{X_3}\right) \cdot S\,(t+1) \tag{3.5}$$

$$S\,(t) = 1 - \frac{ratio\,(t) - \zeta}{abs\,(ratio\,(t) - \zeta)} \cdot \frac{t}{MaxIter} \cdot (ratio\,(t) + 0.02)^{\frac{1}{ratio(t)^2 + 0.01}} \tag{3.6}$$

where $k\,(t)$ represents the number of search agents with improved fitness in the t iteration, $SearchAgents$ represents the number of all search agents, $ratio\,(t+1)$ represents the evolutionary success rate of the $t+1$ generation, and $S\,(t)$ represents the step of the t generation.

In Eq (3.2), the concept of evolutionary success rate (*ratio*) is proposed. The evolutionary success rate refers to the ratio of search agents with improved fitness in the previous iteration to the total number of search agents. Through the adaptive adjustment of the wolf pack's step size based on the evolutionary success rate, the algorithm can better handle different situations. When the ratio is less than the threshold value ζ ($\zeta = 0.67$), the evolutionary success rate of the wolf pack is relatively low, indicating that the algorithm may be trapped in a local optimum. Under such circumstances, a larger step size is obtained by subtracting a negative number from 1, which is used to enhance the algorithm's ability to escape the local optima. When the ratio is greater than or equal to the threshold value ζ, the evolutionary success rate of the wolf pack is high, indicating that most wolves have found better positions. This demonstrates that the current search method aligns with the optimization process. Therefore, by subtracting a large positive number from 1, the step size of the grey wolves is reduced, which enhances their local exploitation ability and allows for more precise development. In addition, this article also takes into account the impact of iteration times on step size. In the early stages of iteration, *t* is small, so the step size is large, and the wolf pack tends to conduct global exploration. In the later stages of iteration, *t* is large, so the step size is small, and the wolf pack tends to focus on local exploitation.

3.4. Position update strategy based on dual convergence factors

In the traditional grey wolf optimization algorithm, the positions of the wolf pack are completely guided by alpha, delta, and omega wolves. However, as the number of iterations increases, the wolf pack tends to concentrate in a limited region. This phenomenon significantly heightens the risk of falling into local optima and makes it difficult to jump out of local optima when facing complex problems. The randomness of evolutionary algorithms leads most evolutionary algorithms to be black box optimizers, and we cannot accurately judge when the algorithm is exploring the search space when it is developing, or whether it has fallen into local optima. Therefore, when the evolutionary success rate of the algorithm is low, it may be exploring the search space or may have fallen into local optima. When the algorithm is in the exploration stage, randomly generated positions can help the algorithm explore a broader search space. When the algorithm falls into local optima, randomly generated positions can increase population diversity and help the algorithm jump out of local optima. In both cases, randomly generated positions can provide effective assistance. By adding a convergence factor that decreases with the number of iterations to the randomly generated position, we prioritize exploration in the early stages and increase the ability to jump out of local optima in the later stages. The global optimal position can better guide the search direction of search agents. When the evolutionary success rate is low, the algorithm may have fallen into local optima, so we should reduce the influence of the global optimal position, and vice versa. Therefore, this article proposes a new position update strategy that adds convergence factors to both the global optimal position and randomly generated positions. The equation is as follows:

$$\left\{\begin{array}{l} \overrightarrow{X}(t+1) = \left(1 + e^{ratio(t)}\right) \cdot \overrightarrow{X_p} - e^{\left(-4 \cdot \frac{t^2}{MaxIter^2}\right)} \cdot \left(\left(\overrightarrow{ub} - \overrightarrow{lb}\right) \cdot \overrightarrow{r_6} + \overrightarrow{lb}\right),\ r7 < 0.5 \\ \overrightarrow{X}(t+1) = \left(1 + e^{ratio(t)}\right) \cdot \overrightarrow{X_p} + e^{\left(-4 \cdot \frac{t^2}{MaxIter^2}\right)} \cdot \left(\left(\overrightarrow{ub} - \overrightarrow{lb}\right) \cdot \overrightarrow{r_6} + \overrightarrow{lb}\right),\ r7 > 0.5 \end{array}\right\} \tag{3.7}$$

where $\overrightarrow{r_6}$ is a random vector in [0,1], $\overrightarrow{ub}$ and $\overrightarrow{lb}$ are the lower and upper bounds, respectively, and r_7 is a random value in [0,1].

In the new position update strategy, we changed the strategy of using the three best wolves to guide the evolution direction of the algorithm to using both the global optimal position and randomly generated positions as learning samples. The global optimal position as a learning sample ensures that the search agents evolve in the correct direction, while adding randomly generated positions can increase population diversity, expand the search range of the algorithm, and greatly enhance the ability to jump out of local optima. Second, we utilize the *ratio* to control the inertia weight of the global optimal position, ensuring that the inertia weight of the global optimal position is always greater than 1 to ensure its influence. Additionally, we use an exponential function e^x to amplify the influence of the global optimal position. When the *ratio* is small, the inertia weight of the global optimal position is small, increasing the influence of randomly generated positions and improving the ability to jump out of local optima. Finally, in the early stages of the search, the algorithm should focus on exploration to ensure that search agents explore the search space as much as possible. Therefore, the inertia weight of randomly generated positions is relatively large in the early stages. When x is linearly increasing on the interval [0,1], e^{-4x^2} is nonlinearly decreasing. On the interval [0,1], the function e^{-4x^2} exhibits convexity in the initial segment where the second derivative is greater than 0, indicating a slower rate of decrease. In the later segment, where the second derivative is less than 0, the function demonstrates concavity, indicating a faster rate of decrease. As the number of iterations increases, the inertia weight of randomly generated positions decreases nonlinearly; it decreases slowly in the early stages to maintain a large inertia weight to explore the search space, and it decreases quickly in the later stages while not affecting the exploitation of the algorithm in later stages to provide a possibility for jumping out of local optima.

3.5. Theoretical convergence analysis

Based on the Eqs (3.2) and (3.5), we can derive the position update value of the j-th dimension for the i-th wolf as follows:

$$\begin{aligned}
x_{ij}^{t+1} &= w_1 s_t \left[x_{\alpha j} - (2\alpha_t r_{11} - \alpha_t)\left|2r_{12}x_{\alpha j} - x_{ij}^t\right| spiral_1\right] \\
&+ w_2 s_t \left[x_{\beta j} - (2\alpha_t r_{21} - \alpha_t)\left|2r_{22}x_{\beta j} - x_{ij}^t\right| spiral_2\right] \\
&+ w_3 s_t \left[x_{\delta j} - (2\alpha_t r_{31} - \alpha_t)\left|2r_{32}x_{\delta j} - x_{ij}^t\right| spiral_3\right] \\
&= \left(w_1 x_{\alpha j} + w_2 x_{\beta j} + w_3 x_{\delta j}\right) s_t - a_t \left[(2r_{11} - 1)\left|2r_{12}x_{\alpha j} - x_{ij}^t\right| spiral_1\right. \\
&\left.(2r_{21} - 1)\left|2r_{22}x_{\beta j} - x_{ij}^t\right| spiral_2 + (2r_{31} - 1)\left|2r_{32}x_{\delta j} - x_{ij}^t\right| spiral_3\right]
\end{aligned} \tag{3.8}$$

where x_{ij}^{t+1} represents the value of the j-th dimension position of the ith wolf in the next iteration, x_{ij}^t represents the value of the j-th dimension position of the ith wolf in the current iteration, $x_{\alpha j}, x_{\beta j}, x_{\delta j}$ represents the value of the j-th dimension position of the three best wolves in the current iteration, α_t is the value of α in the current iteration, which decreases from 2 to 0 as the number of iterations increases, s_t is the step length in the current iteration, and $r_{11}, r_{12}, r_{21}, r_{22}, r_{31}, r_{32}$ is a random value in [0,1].

In ASGWO, as the number of iterations increases, α_t gradually approaches 0. When the number of iterations approaches its maximum value, the impact of the second term in Eq (3.8) on the x_{ij}^{t+1} position can be ignored. At this time, s_t also approaches infinity with 1. Assuming that the positions of the three leader wolves remain unchanged, x_{ij}^{t+1} approaches a constant value, so ASGWO has convergence.

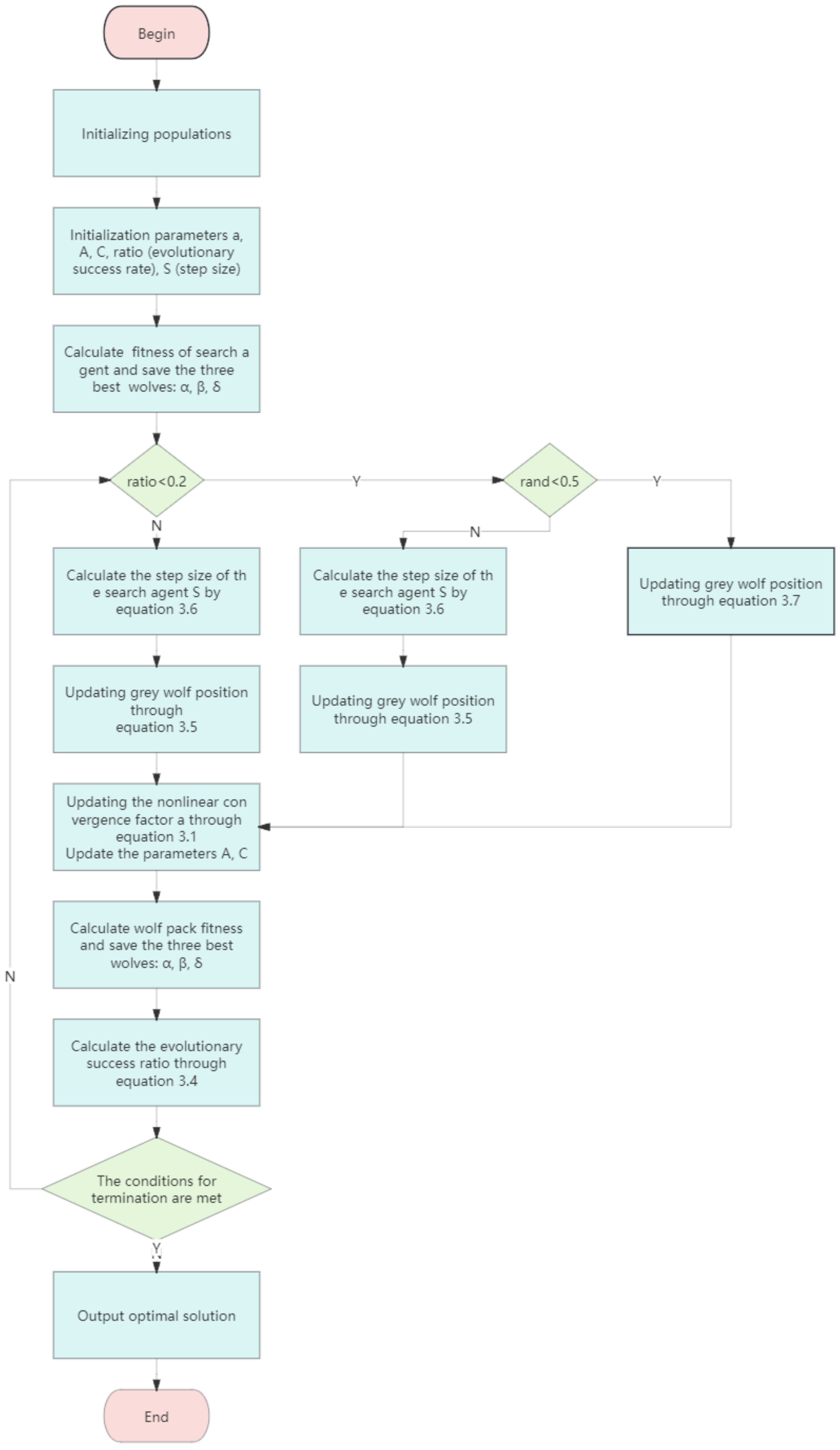
Begin
Initializing populations
Initialization parameters a, A, C, ratio (evolutionary success rate), S (step size)
Calculate fitness of search a gent and save the three best wolves: α, β, δ
ratio<0.2
Y
rand<0.5
Y
N
N
Calculate the step size of th e search agent S by equation 3.6
Calculate the step size of th e search agent S by equation 3.6
Updating grey wolf position through equation 3.7
Updating grey wolf position through equation 3.5
Updating grey wolf position through equation 3.5
Updating the nonlinear con vergence factor a through equation 3.1
Update the parameters A, C
Calculate wolf pack fitness and save the three best wolves: α, β, δ
N
Calculate the evolutionary success ratio through equation 3.4
The conditions for termination are met
Y
Output optimal solution
End

Figure 3. Flowchart of the ASGWO algorithm.

4. Experimental verification and analysis

4.1. Benchmarking functions and testing environment

To evaluate the performance of ASGWO, we used two sets of test functions from the literature [22] to benchmark the algorithm's exploration and exploitation. The first set of test functions are the classic unimodal functions (f1–f7) in Table 1, which have only one global optimal value and do not risk falling into local minima. They are used to test the algorithm's exploitation. The second set of test functions are the common multimodal functions (f8–f13) in Table 2 and fixed-dimension multimodal benchmark functions (f14–f23) in Table 3, which have many local minima and the algorithm is highly likely to fall into local optima. They are used to test the algorithm's ability to jump out of local optima and examine exploration [23]. In Tables 1–3, the third column Dim, represents the dimension of the benchmark function, the fourth column Range, represents the upper and lower limits of the benchmark function, and the fifth column fmin, represents the global minimum point of the benchmark function.

All encoding in this article was implemented on a Windows - 10 platform using Python 3.8 on a computer with an Intel(R) Core(TM) i5-8300H CPU processor and 8GB of memory.

4.2. Composition with GWO and tranditional algorithm

In assessing ASGWO's convergence accuracy, convergence speed, population diversity, and capability to evade local optima, we juxtaposed its optimization efficacy on both unimodal and multimodal test functions with the native GWO algorithm, alongside two classical algorithms: PSO [2] and WOA [3]. The parameter settings for the algorithms are recorded in Table 4. For different benchmark functions, the four algorithms were independently run 30 times, with an iteration number of 500 per independent run and a population size of 20. The average and variance were taken to generate statistical results. The experimental results for unimodal, multimodal, and fixed-dimension multimodal benchmark functions are shown in Tables 3, 5, and 6, respectively.

Table 1. Unimodal benchmark functions.

Function	Dim	Range	fmin
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	30	[−10,10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j^2\right)^2$	30	[−100,100]	0
$f_4(x) = \max_i \{\lvert x_i\rvert, 1 \le i \le n\}$	30	[−100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	[−30,30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[−1.28,1.28]	0

Table 2. Multimodal benchmark functions.

Function	Dim	Range	fmin
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[−500,500]	0
$f_9(x) = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	[−5.12,5.12]	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[−32,32]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600,600]	0
$f_{12} = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{Bmatrix} k(x_i - a)^m \; x_i > a \\ 0 \; -a < x_i < a \\ k(-x_i - a)^m \; x_i < -a \end{Bmatrix}$	30	[−50,50]	0
$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + \right.$ $\left.(x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[−50,50]	0

Table 3. Fixed-dimension multimodal benchmark functions.

Function	Dim	Range	fmin
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})}\right)^{-1}$	2	[−65,65]	1
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[−5,5]	0.0003
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[−5,5]	0.398
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times$ $\left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2	[−2,2]	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	3	[−1,3]	−3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	6	[0,1]	−3.32
$f_{21}(x) = -\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.1532
$f_{22}(x) = -\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.4028
$f_{23}(x) = -\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.5363

Table 4. The parameter settings of the four algorithms.

Algorithm	Parameters
GWO	a linearly decreased over iterations from 2 to 0
PSO	$\omega = 1$, $c1$=2, $c2 = 2$
WOA	a linearly decreased over iterations from 2 to 0
ASGWO	a decreased from 2 to 0 unlinearly, $\zeta = 0.67$

Table 5. The results of unimodal benchmark functions.

Function	GWO		PSO		WOA		ASGWO	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f1	2.42×10^{-26}	3.07×10^{-26}	5.89	5.27	5.28×10^{-7}	1.58×10^{-6}	**0.00**	0.00
f2	4.08×10^{-16}	2.71×10^{-16}	8.85	8.00	2.42×10^{-10}	6.57×10^{-10}	9.1×10^{-243}	6.7×10^{-243}
f3	5.89×10^{-4}	1.62×10^{-2}	22.4	7.83	2.14×10^{-2}	3.60×10^{-2}	**0.00**	0.00
f4	2.83×10^{-5}	1.86×10^{-5}	1.24	0.398	1.27×10^{-2}	2.59×10^{-2}	1.1×10^{-201}	5.7×10^{-201}
f5	27.3	0.813	2.36×10^{2}	1.64×10^{2}	28.6	0.330	**26.3**	0.314
f6	1.37	0.492	9.26	3.25	22.8	53.2	4.39×10^{-5}	1.66×10^{-5}
f7	3.65×10^{-3}	1.52×10^{-3}	1.73×10^{2}	53.1	8.56×10^{-2}	0.177	3.03×10^{-3}	4.71×10^{-4}

Table 6. The results of multimodal benchmark functions.

Function	GWO		PSO		WOA		ASGWO	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f8	-6.2×10^{3}	6.51×10^{2}	-5.2×10^{3}	7.00×10^{2}	-3.3×10^{3}	2.87×10^{2}	-7.0×10^{3}	4.52×10^{2}
f9	13.4	10.6	1.73×10^{2}	22.4	9.42×10^{-12}	2.74×10^{-11}	**0.00**	0.00
f10	1.38×10^{-13}	2.52×10^{-14}	2.78	0.448	1.13×10^{-8}	2.96×10^{-8}	1.11×10^{-14}	2.91×10^{-15}
f11	5.81×10^{-3}	8.93×10^{-3}	0.650	0.174	2.22×10^{-16}	3.71×10^{-13}	**0.00**	0.00
f12	6.86×10^{-2}	5.72×10^{-2}	0.839	0.405	0.868	0.271	1.40×10^{-2}	9.56×10^{-3}
f13	0.632	0.244	1.52	0.757	2.36	0.149	3.34×10^{-5}	1.27×10^{-5}

Table 7. The results of fixed-dimension multimodal benchmark functions.

Function	GWO		PSO		WOA		ASGWO	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f14	5.01	4.27	1.13	0.302	2.86	0.971	**1.10**	0.288
f15	8.38×10^{-3}	9.77×10^{-3}	1.04×10^{-2}	8.53×10^{-3}	3.94×10^{-3}	3.25×10^{-3}	4.53×10^{-3}	7.91×10^{-3}
f16	-1.0	2.99×10^{-8}	0.861	0.921	0.640	0.326	-1.0	5.10×10^{-8}
f17	0.397	6.30×10^{-5}	0.861	0.921	0.640	0.326	**0.397**	5.10×10^{-8}
f18	**3.00**	1.46×10^{-5}	3.02	1.84×10^{-2}	4.20	2.01	**3.00**	5.04×10^{-5}
f19	-3.8	3.47×10^{-3}	-3.8	2.98×10^{-3}	-3.7	3.20×10^{-2}	-3.8	3.15×10^{-3}
f20	-3.2	9.32×10^{-2}	-3.0	0.271	-2.5	0.608	-3.2	4.76×10^{-2}
f21	-8.0	2.46	-8.1	1.62	-2.3	1.91	-9.0	2.00
f22	-7.6	3.16	-6.2	1.76	-1.9	1.41	-8.6	2.97
f23	-10	3.20×10^{-3}	-6.7	1.36	-1.8	1.57	-10	1.61×10^{-5}

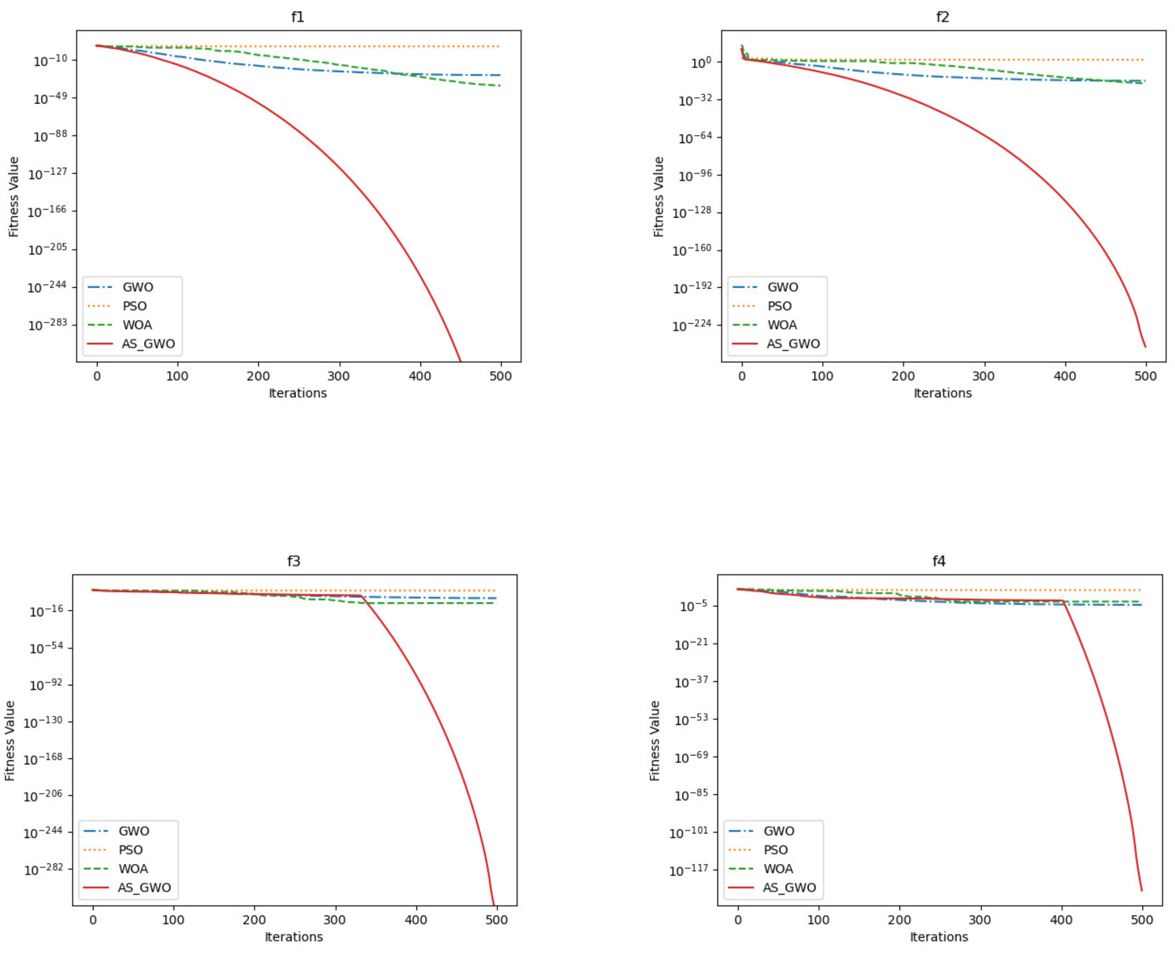

Figure 4. Exponential convergence curve of f1–f4.

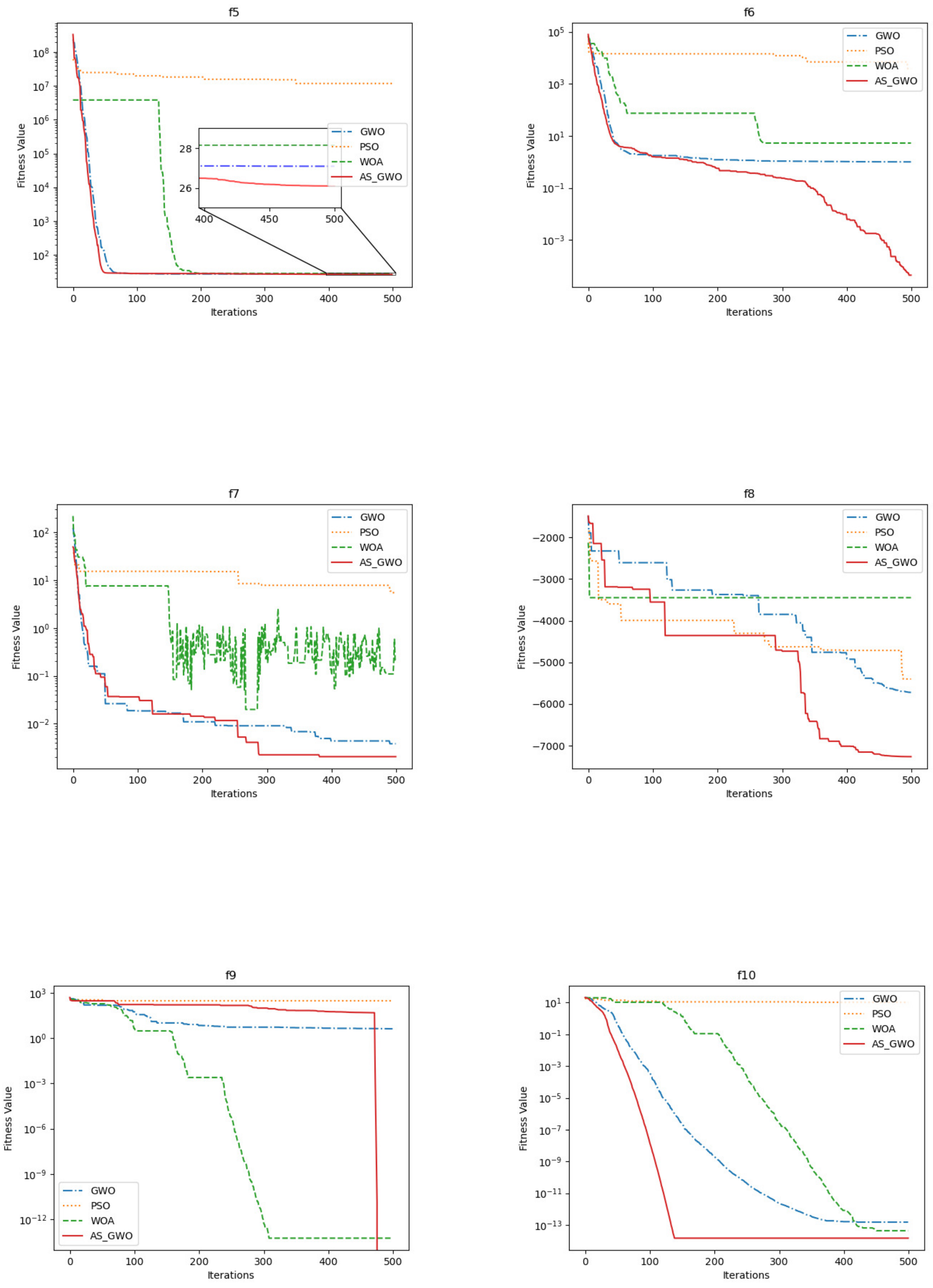

Figure 5. Exponential convergence curve of f5–f10.

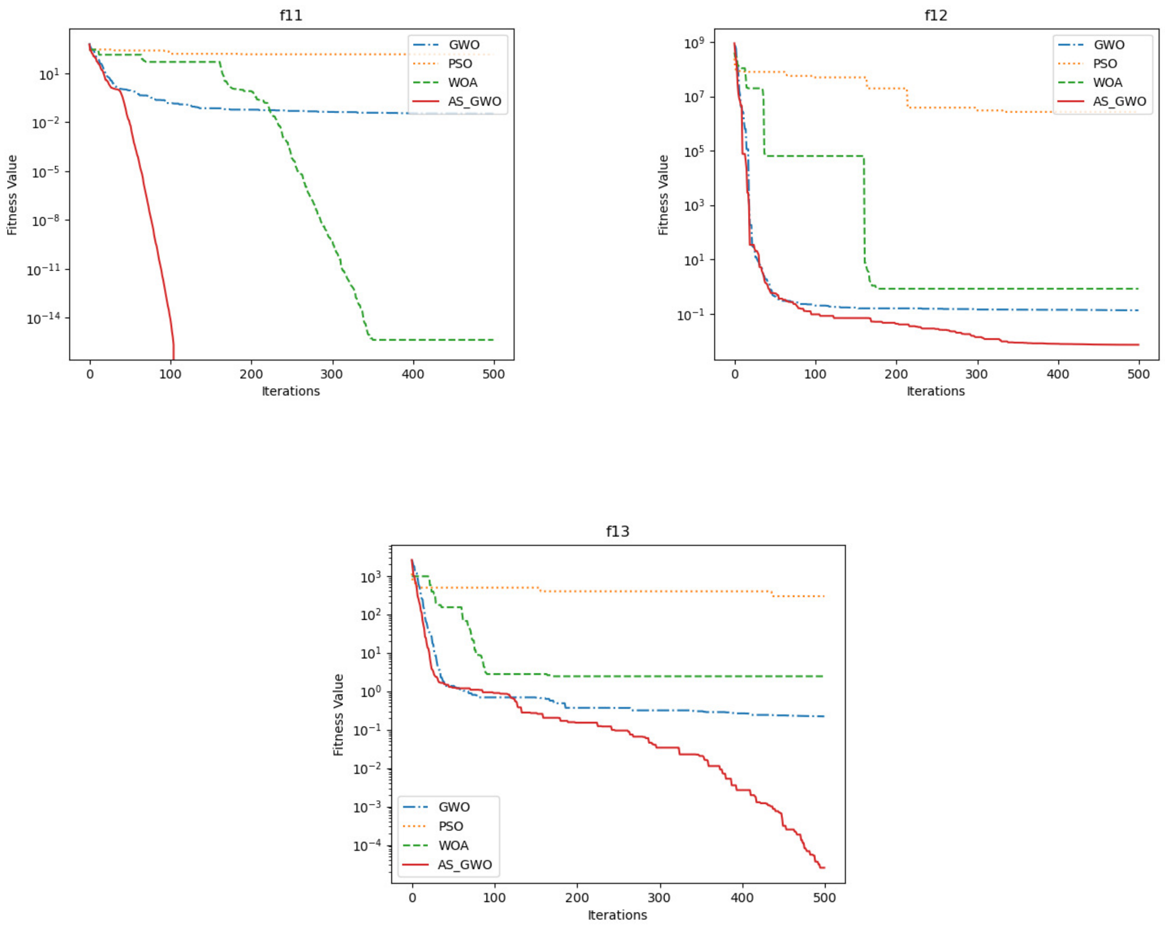

Figure 6. Exponential convergence curve of f11–f13.

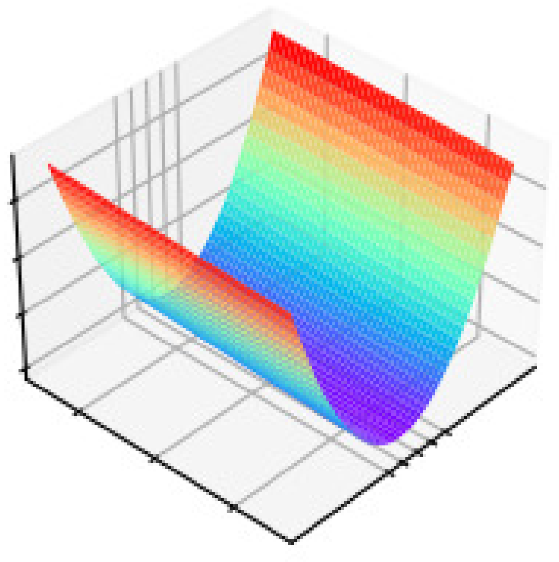

Figure 7. 2-D version of f5.

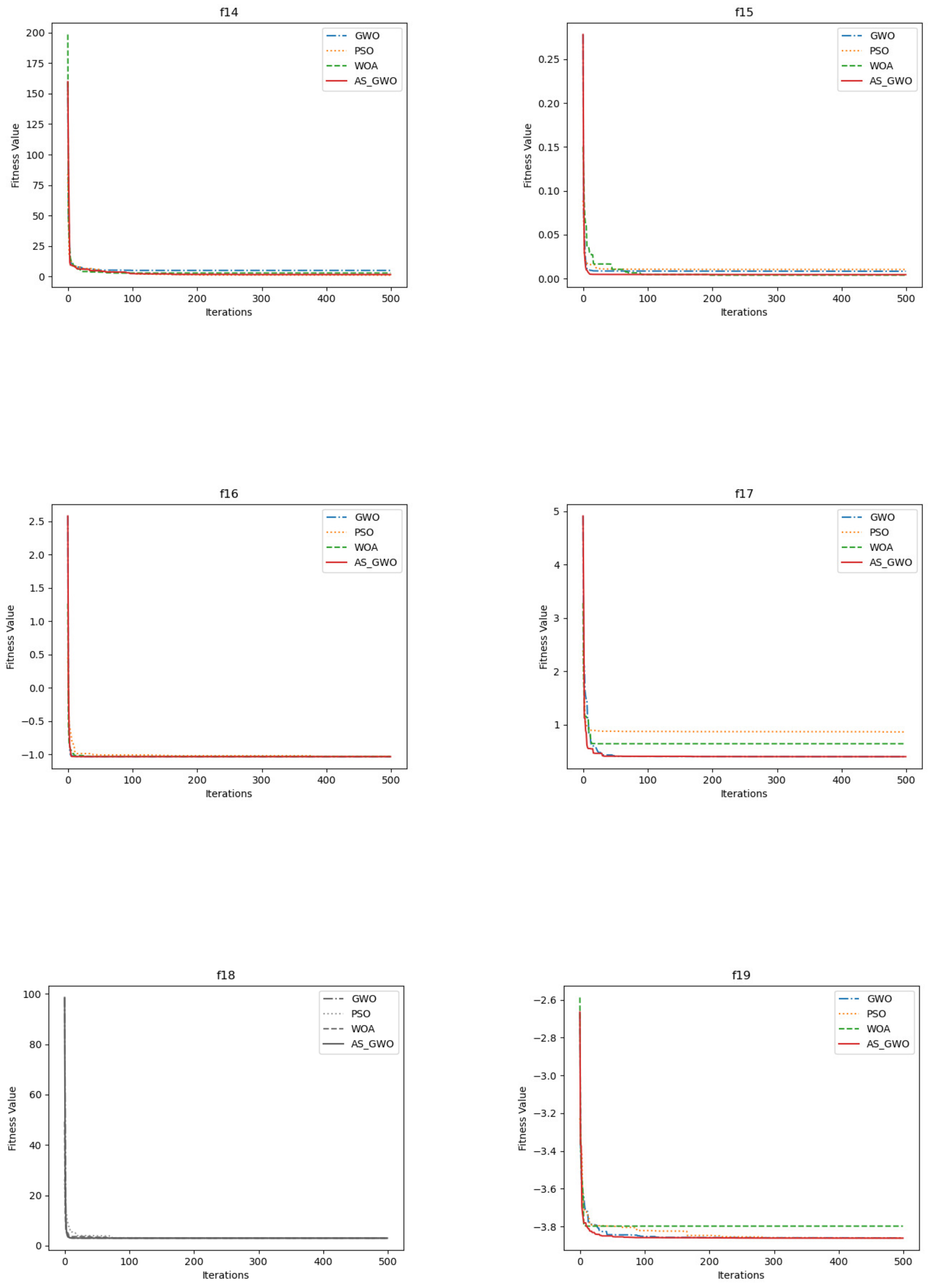

Figure 8. Actual convergence curve of f14–f19.

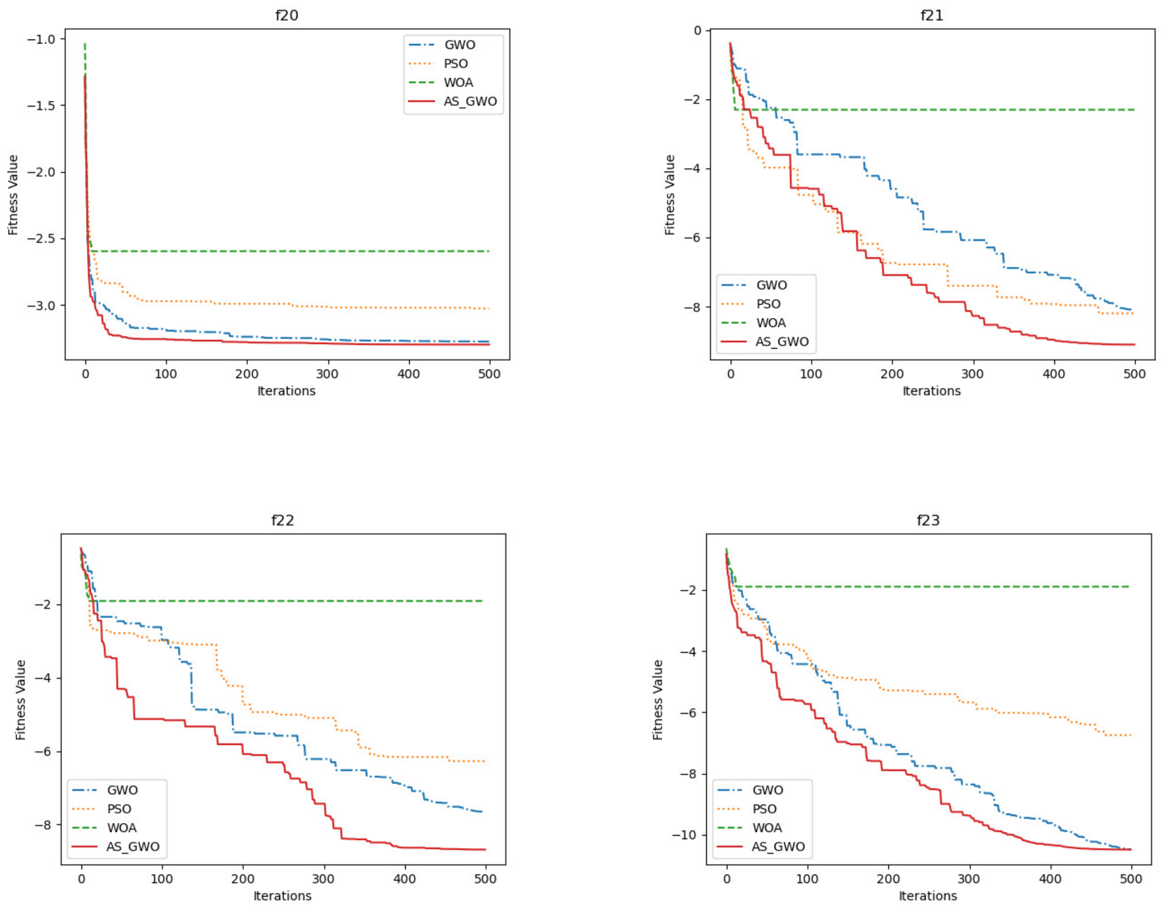

Figure 9. Actual convergence curve of f20–f23.

4.2.1. Unimodal function analysis

According to the experimental results of Table 3 for the unimodal test functions, we can observe that ASGWO exhibits superior performance compared to GWO, PSO, and WOA. First, in the test functions f1 and f3, ASGWO found the global optimal value, while the other three algorithms were still distant from the global optimal value. Second, in the test functions f2, f4, f6, and f7, although ASGWO did not find the global optimal value, it still demonstrated a significant improvement in convergence accuracy compared to the original GWO, PSO, and WOA. Finally, as shown in Figure 7, the contour lines of function f5 form a parabolic shape, and the global optimal value lies in the valley of this parabolic shape. While it may be easy for algorithms to find this valley, convergence to the global optimal value is extremely challenging due to the slow gradient change within this narrow valley. Therefore, the performance improvement of ASGWO on the function f5 was not as significant as expected, but it still outperformed GWO, PSO, and WOA. Additionally, as shown in Table 1, function f6 is a step function, which is characterized by plateaus and discontinuity. Since GWO, PSO, and WOA performing searches within local neighborhoods, all the points within the local neighborhood will have the same fitness value except for a few boundaries between plateaus, it is difficult for them to move from the current plateau to a lower plateau. However, ASGWO's adaptive step size can help the algorithm produce longer jumps with a higher probability, making it easier for ASGWO to move towards lower plateaus. As shown in Figures 4 and 5, ASGWO's convergence speed far exceeded the other algorithms. Finally, the experimental results of Table 3 indicate that compared to the other three algorithms, ASGWO has a smaller standard deviation, representing more stable convergence and stronger robustness.

In summary, ASGWO has significantly improved the convergence accuracy, convergence speed, and robustness of unimodal test functions. This is because ASGWO improves the local development ability of the algorithm by rapidly decreasing the nonlinear convergence factor in the later stages, and utilizes more path information through the spiral to improve the local development ability by making the spiral smaller in the later stages, thereby improving the convergence accuracy. In addition, the dynamic spiral and adaptive step size also significantly contribute to the improvement of convergence speed.

4.2.2. Multimodal function analysis

The experimental results of Table 5 indicate that ASGWO still performs better than GWO, PSO, and WOA on multimodal test functions. First, in the test functions f9 and f11, ASGWO found the global optimal value. In contrast, the original GWO, PSO, and WOA had significant differences in convergence accuracy. The test functions f9 and f11 have the characteristics of highly multimodal and regularly distributed minimum positions, suggesting that ASGWO performs well on multimodal functions with regularly distributed minimum positions. Additionally, from the convergence curve of function f9 in Figure 5, we can observe that, even when ASGWO gets trapped in a local optimum in the later stages of the algorithm, it still has the ability to escape from the local optimum and find the global optimal value. Then, for the remaining functions f8, f10, f12, and f13, ASGWO did not find the global optimal value. However, the final convergence result of ASGWO is still superior to the other three algorithms. Therefore, ASGWO has a significant improvement in the convergence results on multimodal functions with many local minima. This is because the nonlinear convergence factor decreases slowly in the early stage, allowing ASGWO to fully explore the search space and lay a solid foundation for avoiding premature convergence. Due to the similarity of function f12 and function f5, the improvement of ASGWO on function f12 did not meet our expectations. From Figures 6 and 7, we can see that because of the complexity of multimodal functions, algorithms may still get trapped in local optima. Therefore, we use the evolution success rate to assess the state of the algorithm. When the algorithm gets trapped in a local optimum, increasing the step size can help it escape from the local optimum. Additionally, new position update strategies can also improve population diversity and help the algorithm escape from local optima. Furthermore, function f8 is a typical deception problem: there is only one global optimal point, which is far away from the local minima. Getting trapped in a local optimum is difficult to escape. However, as shown in Figure 5, ASGWO still demonstrates an impressive ability to escape from local optima on function f8. Finally, from Figures 5–8, we can see that ASGWO still exhibits good convergence speed on multi-peak functions.

On the experimental results of the Table 7 fixed-dimension multimodal benchmark functions, ASGWO has a slight gap with WOA on f15, but has significant advantages on other functions. From Figure 9, it can be seen that ASGWO has a significant improvement in convergence speed on complex functions.

4.2.3. Convergence analysis

As illustrated in Figure 4, ASGWO places emphasis on exploring the search space during the early stages, leading to swift convergence in the subsequent phases. Owing to the GWO's linearly decreasing convergence factor, the algorithm encounters inadequate exploration in the initial phases and gradual convergence in the later stages. To mitigate this, we introduced a modification, transitioning it to a piecewise nonlinear convergence factor. Furthermore, during the exploitation stage, a preference for a smaller step size is evident in ASGWO to ensure precise convergence, as opposed to larger step sizes that might induce oscillations and impact convergence speed. This is facilitated through ASGWO's incorporation of a dynamic self-learning step size, computed based on the current iteration number and population evolution success rate. This adaptation is reflected in the substantial enhancement of convergence speed, as evidenced in Figures 5, 6, and 9.

The original strategy of the GWO algorithm involves the wolf pack consistently converging towards the best three wolves, leading to premature convergence, diminished population diversity, and a propensity to be ensnared in local optima. As depicted in Figures 5 and 6, ASGWO maintains convergence even when conventional algorithms succumb to local optima entrapment. This attribute is credited to the dynamic logarithmic spiral, which empowers the algorithm to glean more information along the path, thereby enriching population diversity. Moreover, the updated position update equation introduces a dynamic influence factor, endowing more significant influence to randomly generated positions during the initial stages, further bolstering population diversity. In Figures 5 and 9, the descending zigzag shape evident in ASGWO's convergence curves suggests that, utilizing the dynamic self-learning step size, ASGWO adapts its step size to be larger during periods of low evolution success rate. This strategic adjustment enhances the algorithm's capacity to evade local optima. In summary, these refinements in ASGWO culminate in enhanced optimization performance, convergence speed, and population diversity.

4.3. Composition with GWO Variants

In Section 4.2, wherein ASGWO was juxtaposed with traditional algorithms, it exhibited notable superiority. To further substantiate ASGWO's optimization prowess, we opted to assess it against two novel variants of the GWO algorithm, specifically SOGWO [24] and EOGWO [25]. SOGWO utilizes Spearman's correlation coefficient to select certain dimensions of the ω wolves for opposition learning, thus avoiding unnecessary exploration and enabling rapid convergence without compromising the probability of finding the optimal solution. EOGWO performs a simplex based opposition on all the wolves. Instead of taking the upper and lower limits of the function, opposition is done using the limits of all the wolves. For different benchmark functions, the three algorithms were independently run 25 times each with a maximum iteration of 1000 and a population size of 50. The average value and variance were calculated to generate statistical results, as shown in Table 8.

From the experimental results in Table 8, we can see that ASGWO converges to the global optimum point on 30-dimensional unimodal functions f1–f4, 30-dimensional multimodal functions f9, and f11, and fixed-dimension multimodal benchmark functions f16–f20, while SOGWO and EOGWO only converge to the global optimum point on function f9. In addition, ASGWO also has good performance on functions f5, f6, f8, f12, and f13, with convergence accuracy far exceeding SOGWO and EOGWO. Finally, ASGWO performs slightly worse than SOGWO and EOGWO on function f7, f10, and f15, but the error is within a reasonable range. Therefore, compared with new GWO variants, ASGWO still has good optimization performance.

Table 8. The result of benchmark functions.

Function	SOGWO		EOGWO		ASGWO	
	Mean	Std	Mean	Std	Mean	Std
f1	6.04×10^{-77}	1.48×10^{-76}	2.81×10^{-71}	8.46×10^{-71}	**0.00**	0.00
f2	1.17×10^{-44}	1.34×10^{-44}	4.31×10^{-42}	7.87×10^{-42}	**0.00**	0.00
f3	5.39×10^{-22}	2.59×10^{-21}	1.52×10^{-20}	4.02×10^{-20}	**0.00**	0.00
f4	7.08×10^{-21}	1.51×10^{-19}	8.06×10^{-19}	1.11×10^{-18}	**0.00**	0.00
f5	26.4	0.762	26.3	0.7364	**25.2**	0.663
f6	0.282	0.247	0.3290	0.245	1.06×10^{-6}	2.37×10^{-7}
f7	4.93×10^{-4}	2.71×10^{-4}	6.07×10^{-4}	4.32×10^{-4}	1.04×10^{-3}	6.49×10^{-4}
f8	-6.5×10^{3}	8.02×10^{2}	-6.27×10^{3}	7.71×10^{2}	-7.31×10^{3}	8.67×10^{2}
f9	0.00	0.00	0.00	0.00	**0.00**	0.00
f10	8.88×10^{-16}	0.00	1.40×10^{-14}	3.20×10^{-15}	1.36×10^{-14}	2.71×10^{-15}
f11	0.00	0.00	1.68×10^{-3}	4.80×10^{-3}	**0.00**	0.00
f12	5.60×10^{-2}	1.42×10^{-5}	2.29×10^{-2}	1.85×10^{-2}	3.91×10^{-3}	3.16×10^{-3}
f13	0.352	0.128	0.257	0.164	1.27×10^{-6}	4.73×10^{-7}
f14	3.40	3.72	3.82	3.86	**0.998**	8.24×10^{-13}
f15	2.38×10^{-3}	6.02×10^{-3}	5.24×10^{-3}	8.68×10^{-3}	4.31×10^{-3}	7.01×10^{-3}
f16	−1.03	3.75×10^{-9}	−1.02	3.45×10^{-9}	−1.03	2.42×10^{-11}
f17	0.397	4.85×10^{-7}	**0.398**	4.82×10^{-7}	**0.398**	1.68×10^{-8}
f18	**3.00**	4.63×10^{-6}	**3.00**	3.60×10^{-6}	**3.00**	3.57×10^{-6}
f19	−3.86	2.71×10^{-3}	−3.86	2.36×10^{-3}	−3.86	1.03×10^{-6}
f20	−3.26	7.37×10^{-2}	−3.27	7.55×10^{-2}	−3.32	2.93×10^{-8}
f21	−9.65	1.50	−9.93	2.07	−9.34	1.40
f22	−10.4	2.65×10^{-4}	−10.2	1.05	−10.6	1.06×10^{-5}
f23	−10.4	0.540	−10.2	1.62	−10.4	1.04×10^{-5}

5. Applications of ASGWO on real engineering problems

ASGWO has exhibited promising outcomes when compared with GWO, PSO, WOA, SOGWO, and EOGWO across 23 test functions. In order to ascertain the efficacy of ASGWO in unfamiliar domains characterized by constraints, we juxtapose it with diverse algorithms on four practical application problems.

5.1. Design of gear train problem

The gear train design problem is an unconstrained discrete design problem in mechanical engineering. It involves arranging and combining multiple gears in a specific way to transmit rotational motion and force from one shaft to another. To simplify the problem, we only consider the gear ratio, which is the most basic factor, as shown in Figure 10. The objective of this problem is to minimize the gear ratio as close as possible to 1/6.931. The gear ratio is defined as the ratio of the angular velocity of the output shaft to the angular velocity of the input shaft. For matching gears, this ratio is inversely proportional to the number of teeth on the input and output gears. The minimum tooth count for each gear is 12, and the maximum tooth count is 60. Treating the number of teeth A(x_1), teeth B(x_2), teeth C(x_3), and teeth D(x_4) as a design variable, reasonable selection and optimization of this variable can be used to achieve better performance of the gear system. Mathematically, the problem is stated as follows:

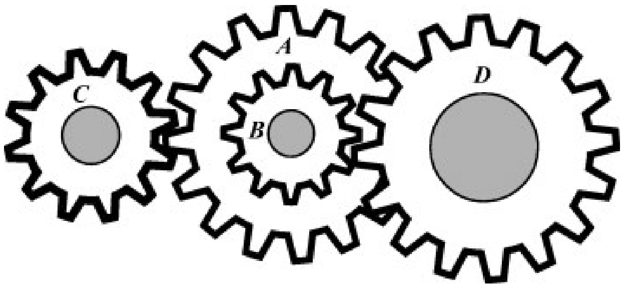

Figure 10. Design of gear train problem.

$$Min\ f(x) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2 s.t.\ 12 \le x_1, x_2, x_3, x_4 \le 60 \tag{5.1}$$

We solve this problem with ASGWO and compare the results to GWO, WOA, K-WOA [26], IWOA [27], GSA-BBO [28], GSO [29], ABC [30], CAB [31], CS [32], FUZZY [33], and MFO [34] in Table 10. K-WOA utilizes K-means clustering to create multiple collaborative search sub-groups based on WOA to explore the search space; IWOA assigns exploration or exploitation to search agents based on their fitness. All the parameters of these algorithms are recorded in Table 9. The results show the average best fitness obtained from 30 independent executions of each algorithm, the standard deviation (SD) of the best fitness obtained from each independent execution, and the optimization parameters (x_1, x_2, x_3, x_4) selected in the best solution of each algorithm. The experimental result of WOA, K-WOA, IWOA, GSA-BBO, GSO, ABC, CAB, CS, FUZZY, and MFO in Table 10 are from the literature [26]. In Table 10, the ASGWO algorithm obtains the best value with transmission ratio (4.14×10^{-15}).

5.2. Design of pressure vessel problem

The objective of Pressure Vessel Design (PVD) is to minimize the total cost related to materials, forming, and welding while fulfilling production requirements, as shown in Figure 11. This engineering problem involves four constraints and four design variables: shell thickness ($T_s = x_1$), head thickness ($T_h = x_2$), inner radius ($R = x_3$), and vessel length ($L = x_4$). The welding cost is divided into vertical welding cost and horizontal welding cost. The estimation method is to multiply the average cost per pound of welding material by the weight of the required welding material, which is $0.6224x_1x_3x_4 + 1.7781x_2x_3^2$. The material and forming costs will be represented by combining the two costs into an average cost per forming operation, which is $3.1661x_1^2x_4 + 19.84x_1^2x_3$. Mathematically, the problem is stated as follows:

Table 9. Parameter settings for gear train problem.

Parameter	Value
same initialization configuration	Population Size is 50, Maxiter is 50000
ASGWO	a decreased from 2 to 0 unlinearly, $\zeta = 0.67$
GWO	a linearly decreased over iterations from 2 to 0
WOA	a linearly decreased over iterations from 2 to 0
IWOA	Scaling factor for beta (0.2, 0.8), DE mutation scheme(DE/best/1/bin), a linearly decreased over iterations from 2 to 0
K-WOA	fixed number of clusters $k = 18$
GSA-BBO	$k = 2$, $I = 1$, $E1 = 1$, $Siv = 4$, $Rnorm = 2$, $Rpower = 1$
GSO	the acceleration constants are 2.05
ABC	Onlooker 50%, employees 50%, acceleration coefficient upper bound (a)= 1, LL = (0.6×dimensions×population)
CAB	$Mbest = 4$, $Hp = 0.2$
CS	$\beta = 1.5$, Discover = 0.25
FUZZY	Nflames = round (Npop - k) (Npop - 1) kmax
MFO	$c1$=2, $c2$ = 2, ω decreased from 0.9 to 0.2

Table 10. The result of gear train design problem.

Algorithm	Optimal solution				Optimal cost	SD
	x_1	x_2	x_3	x_4		
ASGWO	24	18	59	53	4.14×10^{-15}	7.57×10^{-15}
GWO	26	18	60	54	1.49×10^{-14}	2.75×10^{-14}
WOA	16	19	49	43	1.15×10^{-9}	1.39×10^{-9}
IWOA	30	13	51	53	2.39×10^{-9}	2.53×10^{-9}
K-WOA	19	16	43	49	2.70×10^{-12}	0.00
GSA-BBO	16	19	49	43	8.72×10^{-10}	8.38×10^{-10}
GSO	60	29	52	60	0.732	0.00
ABC	16	19	49	43	6.62×10^{-11}	1.65×10^{-10}
CAB	12	12	35	12	0.675	0.180
CS	16	19	43	49	1.47×10^{-10}	2.65×10^{-10}
FUZZY	12	23	33	57	2.57×10^{-3}	4.87×10^{-3}
MFO	19	16	49	43	4.85×10^{-9}	6.90×10^{-9}

$$\begin{aligned}
&Min\ f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&s.t.\ g_1(x) = -x_1 + 0.0193x_3 \leq 0 \\
&g_2(x) = -x_2 + 0.00954x_3 \leq 0 \\
&g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^2 + 1296000 \leq 0 \\
&g_4(x) = x_4 - 240 \leq 0 \\
&1 \times 0.0625 \leq x_1, x_2 \leq 99 \times 0.0625 \\
&10 \leq x_3, x_4 \leq 200
\end{aligned} \tag{5.2}$$

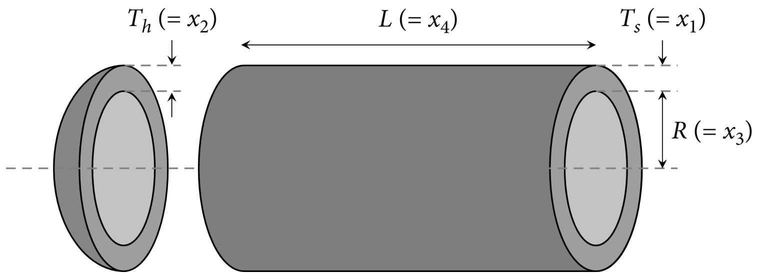

Figure 11. Design of pressure vessel problem.

In order to find the optimal cost, the ASGWO algorithm is implemented 30 times on this problem and the recorded results are shown in Table 12. We obtain the results of GWO, WOA, PSO, GA, SSA, ES [35], SC-GWO [36], mGWO [37], wGWO [38], and chaotic SSA from literature [36], which are also presented in the same table. SC-GWO combines the SCA, which integrates social and cognitive components, with GWO that balances exploration and exploitation. mGWO uses adaptive methods to strike a balance between exploration and exploitation. All the parameters of these algorithms are recorded in the Table 11. From the Table 12, it can be observed that the optimal cost of the proposed algorithm (6010.9908) is better than all other reported algorithms.

Table 11. Parameter settings for pressure vessel problem.

Parameter	Value
same initialization configuration	Population Size is 25, Maxiter is 500
SC-GWO	ω decreased from 0.7 to 0.2
mGWO	$a = 2\left(1 - \frac{t}{Maxiter}\right)$
wGWO	a linearly decreased over iterations from 2 to 0
GA	Crossover Rate = 0.7, Mutation Rate = 0.01
SSA	$c1$ unlinearly decreased over iterations
Chaotic SSA	$c1$ = logistic chaotic map($c1$)
ES	$\sigma = 3.0$, $\mu = 100$, $\lambda = 300$

Table 12. The result of design of pressure vessel problem.

Algorithm	Optimal solution				Optimal cost
	x_1	x_2	x_3	x_4	
ASGWO	0.8327	0.4122	43.1396	168.1458	**6010.9908**
GWO	0.8750	0.4375	44.9807	144.1081	6136.6600
SC-GWO	0.8125	0.4375	42.0984	176.6370	6059.7179
mGWO	0.8125	0.4375	42.0982	176.6386	6059.7359
wGWO	0.8125	0.4375	42.09842	176.637	6059.7207
PSO	0.8125	0.4375	42.0913	176.7465	6061.0777
GA	0.9375	0.5000	48.3290	112.6790	6410.3810
SSA	0.8125	0.4375	42.09836	176.6376	6059.7254
Chaotic SSA	0.8750	0.4375	45.33679	140.2539	6090.527
WOA	0.8125	0.4375	42.0982	176.6389	6059.7410
ES	0.8125	0.437	42.0980	176.6405	6059.7456

Table 13. The result of car crashworthiness problem.

Algorithm	ASGWO	IROA	SMA	HHOCM	ROLGWO	MALO
x_1	0.500041	0.5	0.5	0.50016380	0.5012548	0.5
x_2	1.1345446	1.23105679	1.22739249	1.248612358	1.2455510	1.22810442
x_3	0.5000862	0.5	0.5	0.65955791	0.50004578	0.5
x_4	1.2790514	1.19766142	1.20428741	1.098515362	1.18025396	1.21264054
x_5	0.5002007	0.5	1.20428741	0.757988599	0.50003477	0.5
x_6	1.4999609	1.07429465	1.04185969	0.76726834	1.16588047	1.30804056
x_7	0.5000544	0.5	0.5	0.500055187	0.50008827	0.5
x_8	0.3449606	0.3449999	0.345	0.34310489	0.3448952	0.34499984
x_9	0.3324805	0.3443286	0.3424831	0.19203186	0.2995826	0.28040129
x_{10}	−16.33320	0.9523965	0.2967546	2.89880509	3.5950796	0.42429341
x_{11}	−2.149117	1.0114033	1.1579641	−4.5511746	2.2901802	4.65653809
fmin	**22.871876**	23.188937	23.191021	24.483584	23.222427	23.229404

5.3. Design of car crashworthiness problem

The design of car crashworthiness poses a challenge in the context of car side impact mitigation, aiming to minimize vehicle weight, passenger impact forces, and the average velocity of the V-shaped pillar. This challenge encompasses ten constraints, including limits on abdominal load, pubic force, V-pillar velocity, rib defects, and so on. Additionally, there were eleven design variables that described the thickness of the B-pillar inner panel (x_1), the B-pillar reinforcement (x_2), the floor inner panel thickness (x_3), the crossbeam (x_4), the door beam (x_5), the door belt line reinforcement (x_6), the roof longitudinal beam (x_7), the B-pillar inner panel (x_8), the floor inner panel (x_9), the guardrail height (x_{10}), and the collision position (x_{11}). The optimization problem formula is as follows:

$$
\begin{aligned}
& Min\ f(x) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 \\
& s.t.\ g_1(x) = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} - 1 \leq 0 \\
& g_2(x) = 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} - 32 \leq 0 \\
& g_3(x) = 33.86 + 2.95x_3 + 0.1792x_3 - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} - 9.98x_7x_8 + 22.0x_8x_9 - 32 \leq 0 \\
& g_4(x) = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} - 32 \leq 0 \\
& g_5(x) = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} + 0.08045x_6x_9 \\
& + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} - 0.32 \leq 0 \\
& g_6(x) = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + 0.0208x_3x_8 \\
& + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + 0.00121x_8x_{11} - 0.32 \leq 0 \\
& g_7(x) = 0.74 - 0.61x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 - 0.32 \leq 0 \\
& g_8(x) = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 - 4 \leq 0 \\
& g_9(x) = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} - 9.9 \leq 0 \\
& g_{10}(x) = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 - 15.7 \leq 0 \\
& 0.5 \leq x_1, x_2, x_3, x_4, x_5, x_6, x_7 \leq 1.5 \\
& 0.192 \leq x_8, x_9 \leq 0.345 \\
& -30 \leq x_{10}, x_{11} \leq 30
\end{aligned}
\tag{5.3}
$$

Table 14. Parameter settings for car crashworthiness problem.

Parameter	Value
same initialization configuration	Population Size is 25, Maxiter is 500
IROA	$C = 0.1$; $\alpha \in [-1,9]$; $\mu = 0.499$; $z = 0.07$; $y = 0.1$
SMA	$z = 0.03$
HHOCM	The value of escaping energy decreases from 2 to 0, mutation rate decreases linearly from 1 to 0
ROLGWO	$r3 \in [0, 1]$
MALO	Switch possibility = 0.5

Through the literature [39], the optimal experimental results of IROA [39], SMA [40], HHOCM [41], ROLGWO [42], and MALO [43] in the design of car crashworthiness problem are obtained. IROA introduces an autonomous foraging mechanism, giving each search agent a small chance to randomly search for food or search based on the current food location. ROLGWO proposes a modified parameter "C" strategy to balance exploration and exploitation in GWO. Additionally, a new random opposite-based learning strategy is introduced to help the population escape local optima. All the parameters of these algorithms are recorded in the Table 14. In this article, under the same experimental parameters (500 iterations and 30 search agents), the ASGWO is tested, and the best experimental result is 22.871876, ranking first among these algorithms. Therefore, ASGWO has outstanding advantages in solving the design of the car crashworthiness problem.

5.4. Feature selection

Data mining is currently a highly discussed topic, with the aim of acquiring and processing large datasets to extract actionable knowledge. However, the high dimensionality of feature space poses a significant challenge in data mining, mainly due to the computational complexity involved. Feature selection has emerged as a solution to overcome this challenge. It aims to choose the most relevant subset of features from the original feature set to reduce dimensionality, lower computational costs, and significantly enhance the efficiency of models. Moreover, feature selection can reduce feature redundancy, thereby improving the generalization ability of models. Therefore, feature selection is an indispensable part of the machine learning process, enabling the construction of simpler, more efficient, and more interpretable machine learning models.

This paper considers feature selection as a multi-objective problem: minimizing the number of selected features and maximizing the feature-measure. The goal of feature selection is to either select or not select the most beneficial features, which is a binary problem. However, the positions generated by ASGWO are continuous and cannot be directly applied to feature selection. Therefore, this paper sets the search space of ASGWO to [0, 1] and maps the positions of the standard ASGWO agents to the binary space using the simplest transformation function, as shown in the equation below.

$$x_{ij}^{binary} = \begin{cases} 0 \; if \; x_{ij} \le 0.5 \\ 1 \; if \; x_{ij} > 0.5 \end{cases} \tag{5.4}$$

where x_{ij} represents the numerical value of the position of the j-th dimension of the i-th search agent, while x_{ij}^{binary} represents the numerical value of the position of the j-th dimension of the i-th search agent mapped to the binary space.

Table 15. List of the datasets.

Datasets	No. of attributes	No. of samples
Breast-w	9	699
Credit-g	20	1000
Dermatology	34	366
Glass	9	214
Ionosphere	34	351
Lymphography	18	148
Sonar	60	208

This paper evaluates the performance of ASGWO using a KNN classifier through a ten-fold cross-validation approach on the seven UCI datasets listed in Table 15. In each run, the F-measure value and the number of selected features are recorded, and the averages are taken over ten iterations. These results are then compared with those obtained for BASO, BGA, BPSO, and BGWO from the literature [44], and the comparisons are tabulated in Table 17. To ensure fairness in testing, the same test parameters as those listed in Table 16 are used for all five algorithms.

Table 16. Parameter settings for feature selection.

Parameter	Value
K for KNN	3
Dimension of population	10
Number of iterations	100
Number of runs	10
Acceleration constants in PSO	[2,2]
Inertia w in BPSO	[0.9,0.4]
Parameter A in BGWO	$min = 0, max = 2$

Table 17. The results of feature selections.

Dataset		Breast-w	Credit-g	Dermato	Glass	Ionosph	Lymph.	Sonar
KNN		0.965	0.593	0.873	0.591	0.817	0.712	0.816
	BASO	0.982	0.829	0.988	**0.778**	0.887	0.896	0.892
	BGA	0.983	**0.831**	0.988	0.750	0.887	0.896	0.892
Average F-measure	BPSO	0.981	0.824	0.987	0.753	0.870	0.893	0.880
	BGWO	0.981	0.825	0.989	0.754	0.853	0.868	0.865
	ASGWO	**0.988**	0.733	**0.998**	0.705	**0.958**	**0.955**	**0.969**
	BASO	6.5	11.1	19.7	7.6	11.4	10.8	27.5
	BGA	6.3	10.6	19.4	6.3	11.4	10.2	28.8
Selected feature	BPSO	6.5	**9.9**	20	7.8	**11.1**	9.9	28.9
	BGWO	7.1	13.9	25.6	7.4	11.7	13.3	41.6
	ASGWO	**4.74**	10.04	**17.82**	**4.92**	13.24	**7.6**	**27.46**

By analyzing Table 17, we can observe that the F-measure results of the KNN classifier with feature selection using ASGWO significantly outperforms the direct application of the KNN classifier. Additionally, the number of features is effectively reduced. Therefore, ASGWO can be effectively applied to feature selection, improving classification accuracy and reducing computational complexity. Notably, on the Lymphography and Sonar datasets, ASGWO outperforms BASO, BPSO, and BGWO in terms of F-measure, while also selecting the fewest number of features. On the Breast-w and Dermatology datasets, although ASGWO has only a slight advantage in F-measure, the number of features is significantly reduced by 24 and 30.3%, respectively, significantly improving the efficiency of the classifier. This is due to ASGWO's self-learning ability, which allows it to fully associate the current state with each feature, enhancing its understanding of feature availability. Furthermore, on the Ionosphere dataset, ASGWO achieves the best F-measure, albeit with a slightly higher number of features compared to other algorithms. This tradeoff of slightly increased computational cost for improved F-measure is entirely acceptable. However, on the Credit-g dataset, ASGWO performs poorly due to the significant increase in sample size compared to the increase in feature size. This is because ASGWO's binary mapping is too simple and cannot make good decisions when dealing with low-dimensional features in multi-class classification tasks.

6. Conclusions

Due to the low convergence accuracy, slow convergence speed, and tendency to get trapped in the local optima of the original grey wolf optimizer (GWO), this article proposes an adaptive dynamic self-learning grey wolf optimization to address these issues. First, a nonlinear piecewise convergence factor is proposed to ensure sufficient search and rapid development. Second, a dynamic logarithmic spiral line based on the number of iterations is used to guide the wolves toward the best wolf, expanding the search range in the early iterations and improving population diversity. In the later iterations, the algorithm's local development ability is enhanced to accelerate convergence. Third, a dynamic self-learning step size based on the rational learning of evolution success rate and the number of iterations is introduced to improve algorithm convergence speed. Through self-learning of current information, calculate the appropriate step size for the current algorithm, preventing the step size from being too cautious or aggressive, to avoid algorithm oscillation and the effect of convergence speed. When the algorithm gets trapped in a local optimum, increasing the step size helps the algorithm escape from the local optimum. Finally, a new position update strategy is proposed. Based on the evolution success rate, the original position update strategy and the new position update strategy are selected. The new position update strategy adds a randomly generated search agent as a learning sample. In the early stage of the algorithm, it can help improve population diversity and expand the search range. In the later stage of the algorithm, it can help escape from local optima. The learning samples of the new position update strategy also include the global optimal position to provide effective guidance for the evolution direction. In addition, controlling the influence of two learning samples based on the algorithm's state using dual convergence factors is crucial in the position update stragegy. One convergence factor ensures global optimal leadership, and the other expands exploration in the early stage, and increases the possibility of jumping out of local optima without affecting development in the later stage. The performance of ASGWO was tested on 23 benchmark functions and compared with classical algorithms GWO, PSO, WOA, and new GWO variants: SOGWO, EOGWO. The experimental results showed that ASGWO had higher convergence accuracy, faster convergence rate, and stronger ability to escape local optima compared to both the original GWO and classical algorithms, as well as new variants. In addition, through the results of real engineering problems, we can find that ASGWO also performs better in the unknown search space, which shows the applicability of ASGWO in solving real problems and feature selection. However, on valley test functions where local optimal changes are not obvious, there is still much room for improvement in the convergence accuracy of ASGWO, which will be our future research direction.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors are grateful to the editor and reviewers for their constructive comments and suggestions, which have improved the presentation. This work is financially supported by Youth Fund of Fundamental Research Funds of Jiangnan University, JUSRP122034.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. F. Jiang, L. Wang, L. Bai, An adaptive evolutionary whale optimization algorithm, in *2021 33rd Chinese Control and Decision Conference (CCDC)*, (2021), 4610–4614. https://doi.org/10.1109/CCDC52312.2021.9601898
2. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95-international conference on neural networks*, **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
3. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Software*, **95** (2016), 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
4. S. Mirjalili, The ant lion optimizer, *Adv. Eng. Software*, **83** (2015), 80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
5. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
6. G. M. Komaki, V. Kayvanfar, Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time, *J. Comput. Sci.*, **8** (2015), 109–120. https://doi.org/10.1016/j.jocs.2015.03.011
7. J. Liu, J. Yang, H. Liu, X. Tian, M. Gao, An improved ant colony algorithm for robot path planning, *Soft Comput.*, **21** (2017), 5829–5839. https://doi.org/10.1007/s00500-016-2161-7
8. M. H. Sulaiman, Z. Mustaffa, M. R. Mohamed, O. Aliman, Using the gray wolf optimizer for solving optimal reactive power dispatch problem, *Appl. Soft Comput.*, **32** (2015), 286–292. https://doi.org/10.1016/j.asoc.2015.03.041
9. R. E. Precup, R. C. David, E. M. Petriu, Grey wolf optimizer algorithm-based tuning of fuzzy control systems with reduced parametric sensitivity, *IEEE Trans. Ind. Electron.*, **64** (2016), 527–534. https://doi.org/10.1109/tie.2016.2607698
10. A. K. M. Khairuzzaman, S. Chaudhury, Multilevel thresholding using grey wolf optimizer for image segmentation, *Expert Syst. Appl.*, **86** (2017), 64–76. https://doi.org/10.1016/j.eswa.2017.04.029
11. R. E. Precup, R. C. David, R. C. Roman, A. I. Szedlak-Stinean, E. M. Petriu, Optimal tuning of interval type-2 fuzzy controllers for nonlinear servo systems using Slime Mould Algorithm *Int. J. Syst. Sci.*, **54** (2023), 2941–2956. https://doi.org/10.1080/00207721.2021.1927236
12. S. Saremi, S. Z. Mirjalili, S. M. Mirjalili, Evolutionary population dynamics and grey wolf optimizer, *Neural Comput. Appl.*, **26** (2015), 1257–1263. https://doi.org/10.1007/s00521-014-1806-7
13. C. A. Bojan-Dragos, R. E. Precup, S. Preitl, R. C. Roman, E. L. Hedrea, A. I. Szedlak-Stinean, GWO-based optimal tuning of type-1 and type-2 fuzzy controllers for electromagnetic actuated clutch systems, *IFAC-PapersOnLine*, **54** (2021), 189–194. https://doi.org/10.1016/j.ifacol.2021.10.032

14. S. Wang, Y. Fan, S. Jin, P. Takyi-Aninakwa, C. Fernandez, Improved anti-noise adaptive long short-term memory neural network modeling for the robust remaining useful life prediction of lithium-ion batteries, *Reliab. Eng. Syst. Saf.*, **230** (2023), 108920. https://doi.org/10.1016/j.ress.2022.108920

15. S. Wang, F. Wu, P. Takyi-Aninakwa, C. Fernandez, D. I. Stroe, Q. Huang, Improved singular filtering-Gaussian process regression-long short-term memory model for whole-life-cycle remaining capacity estimation of lithium-ion batteries adaptive to fast aging and multi-current variations, *Energy*, **284** (2023), 128677. https://doi.org/10.1016/j.energy.2023.128677

16. S. Gottam, S. J. Nanda, R. K. Maddila, A CNN-LSTM model trained with grey wolf optimizer for prediction of household power consumption, in *2021 IEEE International Symposium on Smart Electronic Systems (iSES)*, (2021), 355–360. https://doi.org/10.1109/iSES52644.2021.00089

17. W. Long, J. Jiao, X. Liang, M. Tang, An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization, *Eng. Appl. Artif. Intell.*, **68** (2018), 63–80. https://doi.org/10.1016/j.engappai.2017.10.024

18. Z. J. Teng, J. L. Lv, L. W. Guo, An improved hybrid grey wolf optimization algorithm, *Soft Comput.*, **23** (2019), 6617–6631. https://doi.org/10.1007/s00500-018-3310-y

19. A. Kishor, P. K. Singh, Empirical study of grey wolf optimizer, in *Proceedings of Fifth International Conference on Soft Computing for Problem solving*, (2016), 1037–1049.

20. M. Pradhan, P. K. Roy, T. Pal, Oppositional based grey wolf optimization algorithm for economic dispatch problem of power system, *Ain Shams Eng. J.*, **9** (2018), 2015–2025. https://doi.org/10.1016/j.asej.2016.08.023

21. L. Rodriguez, O. Castillo, J. Soria, P. Melin, F. Valdez, C. I. Gonzalez, A fuzzy hierarchical operator in the grey wolf optimizer algorithm, *Appl. Soft Comput.*, **57** (2017), 315–328. https://doi.org/10.1016/j.asoc.2017.03.048

22. J. Xu, F. Yan, O. G. Ala, L. Su, F. Li, Chaotic dynamic weight grey wolf optimizer for numerical function optimization, *J. Intell. Fuzzy Syst.*, **37** (2019), 2367–2384. https://doi.org/10.3233/jifs-182706

23. E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, *Inf. Sci.*, **179** (2009), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

24. S. Dhargupta, M. Ghosh, S. Mirjalili, R. Sarkar, Selective opposition based grey wolf optimization, *Expert Syst. Appl.*, **151** (2020), 113389. https://doi.org/10.1016/j.eswa.2020.113389

25. S. Zhang, Q. Luo, Y. Zhou, Hybrid grey wolf optimizer using elite opposition-based learning strategy and simplex method, *Int. J. Comput. Intell. Appl.*, **16** (2017), 1750012. https://doi.org/10.1007/s13042-022-01537-3

26. M. A. Navarro, D. Oliva, A. Ramos-Michel, D. Zaldivar, B. Morales-Castaneda, M. Perez-Cisneros, An improved multi-population whale optimization algorithm, *Int. J. Mach. Learn. Cybern.*, **13** (2022), 2447–2478. https://doi.org/10.1007/s13042-022-01537-3

27. S. M. Bozorgi, S. Yazdani, IWOA: An improved whale optimization algorithm for optimization problems, *J. Comput. Des. Eng.*, **6** (2019), 243–259. https://doi.org/10.1016/j.jcde.2019.02.002

28. S. A. Rather, N. Sharma, GSA-BBO hybridization algorithm, *Int. J. Adv. Res. Sci. Eng.*, **6** (2017), 596–608.

29. V. Muthiah-Nakarajan, M. M. Noel, Galactic swarm optimization: a new global optimization metaheuristic inspired by galactic motion, *Appl. Soft Comput.*, **38** (2016), 771–787. https://doi.org/10.1016/j.asoc.2015.10.034

30. D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Glob. Optim.*, **39** (2007), 459–471. https://doi.org/10.1007/s10898-007-9149-x

31. E. Cuevas, M. Gonzalez, D. Zaldivar, M. Perez-Cisneros, G. Garcia, An algorithm for global optimization inspired by collective animal behavior, *Discrete Dyn. Nat. Soc.*, **2012** (2012). https://doi.org/10.1155/2012/638275

32. X. S. Yang, S. Deb, Cuckoo search via Lévy flights, in *2009 World congress on nature & biologically inspired computing (NaBIC)*, (2009), 210–214. https://doi.org/10.1109/NABIC.2009.5393690

33. M. A. Diaz-Cortes, E. Cuevas, J. Galvez, O. Camarena, A new metaheuristic optimization methodology based on fuzzy logic, *Appl. Soft Comput.*, **61** (2017), 549–569. https://doi.org/10.1016/j.asoc.2017.08.038

34. S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowl. Based Syst.*, **89** (2015), 228–249. https://doi.org/10.1016/j.knosys.2015.07.006

35. E. Mezura-Montes, C. A. Coello Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, *Int. J. Gener. Syst.*, **37** (2008), 443–473. https://doi.org/10.1080/03081070701303470

36. S. Gupta, K. Deep, H. Moayedi, L. K. Foong, A. Assad, Sine cosine grey wolf optimizer to solve engineering design problems, *Eng. Comput.*, **37** (2021), 3123–3149. https://doi.org/10.1007/s00366-020-00996-y

37. N. Mittal, U. Singh, B. S. Sohi, Modified grey wolf optimizer for global engineering optimization, *Appl. Comput. Intell. Soft Comput.*, **2016** (2016). https://doi.org/10.1155/2016/7950348

38. F. Yan, X. Xu, J. Xu, Grey Wolf Optimizer With a Novel Weighted Distance for Global Optimization, *IEEE Access*, **8** (2020), 120173–120197. https://doi.org/10.1109/ACCESS.2020.3005182

39. R. Zheng, H. M. Jia, L. Abualigah, Q. X. Liu, S. Wang, An improved remora optimization algorithm with autonomous foraging mechanism for global optimization problems, *Math. Biosci. Eng.*, **19** (2022), 3994–4037. https://doi.org/10.3934/mbe.2022184

40. S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: a new method for stochastic optimization, *Future Gener. Comput. Syst*, **111** (2020),300–323. https://doi.org/10.1016/j.future.2020.03.055

41. E. H. Houssein, N. Neggaz, M. E. Hosney, W. M. Mohamed, M. Hassaballah, Enhanced Harris hawks optimization with genetic operators for selection chemical descriptors and compounds activities, *Neural Comput. Appl.*, **33** (2021), 13601–13618. https://doi.org/10.1007/s00521-021-05991-y

42. W. Long, J. Jiao, X. Liang, S. Cai, M. Xu, A random opposition-based learning grey wolf optimizer, *IEEE Access*, **7** (2019), 113810–113825. https://doi.org/10.1109/ACCESS.2019.2934994

43. S. Wang, K. Sun, W. Zhang, H. Jia, Multilevel thresholding using a modified ant lion optimizer with opposition-based learning for color image segmentation, *Math. Biosci. Eng.*, **18** (2021), 3092–3143. https://doi.org/10.3934/mbe.2021155

44. U. KILIC, E. S. ESSIZ, M. K. KELES, Binary anarchic society optimization for feature selection, *Romanian J. Inf. Sci. Technol.*, **26** (2023), 351–364. https://doi.org/10.1080/00207721.2021.1927236