Research Article

Hybrid Genetic Grey Wolf Algorithm for Large-Scale Global Optimization

Qinghua Gu, Xuexian Li, and Song Jiang

School of Management, Xi’an University of Architecture and Technology, Shaanxi 710055, China

Correspondence should be addressed to Song Jiang; jiangsong925@163.com

Received 18 October 2018; Revised 26 December 2018; Accepted 22 January 2019; Published 12 February 2019

Guest Editor: Higinio Mora

Copyright © 2019 Qinghua Gu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Most real-world optimization problems tackle a large number of decision variables, known as Large-Scale Global Optimization (LSGO) problems. In general, the metaheuristic algorithms for solving such problems often suffer from the “curse of dimensionality.” In order to improve the disadvantage of Grey Wolf Optimizer when solving the LSGO problems, three genetic operators are embedded into the standard GWO and a Hybrid Genetic Grey Wolf Algorithm (HGGWA) is proposed. Firstly, the whole population using Opposition-Based Learning strategy is initialized. Secondly, the selection operation is performed by combining elite reservation strategy. Then, the whole population is divided into several subpopulations for cross-operation based on dimensionality reduction and population partition in order to increase the diversity of the population. Finally, the elite individuals in the population are mutated to prevent the algorithm from falling into local optimum. The performance of HGGWA is verified by ten benchmark functions, and the optimization results are compared with WOA, SSA, and ALO. On CEC’2008 LSGO problems, the performance of HGGWA is compared against several state-of-the-art algorithms, CCPSO2, DEwSAcc, MLCC, and EPUS-PSO. Simulation results show that the HGGWA has been greatly improved in convergence accuracy, which proves the effectiveness of HGGWA in solving LSGO problems.

1. Introduction

Large-Scale Global Optimization (LSGO) is widely applied in practical engineering problems, such as large-scale job shop scheduling problem [1], large-scale vehicle routing problems [2], and reactive power optimization of large-scale power system [3]. It is a very important and challenging task in the optimization domain and usually involves thousands of decision variables. As the number of dimensions increases, the complexity of the problem increases exponentially, so it is very difficult to solve. Metaheuristic algorithms are a class of intelligent optimization algorithms inspired by biological activities or physical principles. In recent years, various metaheuristic algorithms such as genetic algorithms, particle swarm optimization algorithms, artificial bee colony algorithms [4], and differential evolution algorithms [5] have been applied to a variety of large-scale global optimization problems. However, due to the existence of “curse of dimensionality”, it is difficult for general metaheuristic algorithms to find the optimal solution of LSGO problems. In order to solve LSGO problems better, many valuable attempts have been made in recent years by using metaheuristic algorithms.

In general, solving large-scale problems can be considered in two ways. First, from the perspective of the problem itself, the solution process can be made more efficient by simplifying the problem. Secondly, from the perspective of the method, the optimal solution to the problem can be solved with greater possibility by improving the performance of the solution algorithm. First of all, in terms of the problem itself, a large number of decision variables are the main cause of the complexity of the problem [6]. In the famous book *A Discourse on Method* [7], Descartes pointed out that it is necessary to study complex problems and decompose them into a number of relatively simple small problems and then solve them one by one. He called it a “divide and conquer” strategy, which is to solve the whole problem by decomposing the original large-scale problem into a set of smaller and simpler subproblems which are more manageable and easier to solve and then solve each subproblem independently. This method based on “divide and conquer” strategy is
also called Cooperative Coevolution (CC), and its effectiveness for solving large-scale optimization problems has been demonstrated in many classical optimization methods. But a major difficulty in applying CC is the choice of a good decomposition strategy, because different decomposition strategies may lead to different optimization effects. Due to the diversity of problem-solving, there may be very large differences between different problems. Therefore, it is necessary to study the structure inherent in the problem and analyze the relationship between the variables in order to find a suitable solution.

In addition, from an algorithmic perspective, two or more distinct methods when combined together in a synergistic manner can enhance the problem-solving capability of the derived hybrid [8]. EAs hybridized with local search algorithms have been successful within function optimization domain. These kinds of EAs are often named as memetic algorithms (MAs) [9]. The optimization performance of the algorithm for large-scale optimization problems is improved by different optimization strategies such as designing new mutation operators, dynamic neighborhood search strategies, multiple classifier [10], and Opposition-Based Learning [11]. Similar to evolutionary computation, the swarm intelligence (SI) is a kind of optimization algorithm inspired by the biological foraging or hunting behavior in nature, which simulates the intelligent behavior of insects, bird groups, ant colonies, or fish schools. Inspired by the grey wolf population hierarchy and hunting behavior, Mirjalili proposed another swarm intelligence optimization algorithm, grey wolf optimization algorithm (GWO) in 2014 [12]. The GWO algorithm has the advantages of less control parameters and fast convergence and it has been applied to various optimization fields, such as medical image fusion [13], multiple input multiple output power systems [14], and job shop scheduling problem [15]. In addition, the proposal of different prediction techniques [16] also provides reference for the study of large-scale optimization problems.

To direct at the large-scale global optimization problems, this paper improves the HGGWA algorithm proposed in literature [17] by adding a parameter nonlinear adjustment strategy, which further improves the global convergence and solution accuracy. The crossover operation of the HGGWA algorithm divides the whole population into several subpopulations by referring to the idea of cooperative coevolution, and then independently evolves each subpopulation separately. More specifically, this paper has the following research objectives:

1. To review the current literature on solving large-scale global optimization problems and to analyze the existing problems in the research
2. To improve the Hybrid Genetic Grey Wolf Algorithm (HGGWA) by adding the nonlinear adjustment strategy of parameter, which further improves the global convergence and solution accuracy for solving large-scale global optimization problems
3. To analyze the global convergence and computational complexity of the improved HGGWA algorithm and to verify the effectiveness of HGGWA for solving large-scale global optimization problems by several different numerical experiments

The remainder of this paper is organized as follows: in Section 2, a review of Large-Scale Global Optimization (LSGO) and various solving strategies is given. Then, Section 3 briefly describes the Grey Wolf Optimizer (GWO). In Section 4, the principles and steps of the improved algorithm (HGGWA) are introduced. Section 5 illustrates and analyzes the experimental results. Finally, the conclusions are drawn in Section 6.

2. Literature Review

2.1. Decomposition-Based CC Strategy. Large-scale global optimization problems are mainly solved in the following two ways. One is Cooperative Coevolution (CC) with problem decomposition strategy. The idea of decomposition was first proposed by Potter and De Jong [18, 19] in 1994. They designed two Cooperative Coevolutionary Genetic Algorithms (CCGA-1, CCGA-2) to improve the global convergence of GA. The CC strategy is to reduce the scale of the high-dimensional problem by dividing the whole solution space into multiple subspaces and then use multiple subpopulations to achieve the cooperative coevolution. Table 1 summarizes proposed major variants of the CC method. It can be divided into the static grouping-based CC methods and the dynamic grouping-based CC methods.

The static grouping-based CC methods divide the population into multiple fixed-scale subpopulations. The Cooperative Coevolutionary Genetic Algorithm (CCGA) proposed by Potter and De Jong is the first attempt to combine the idea of Cooperative Coevolution with metaheuristic algorithms. They decomposed an n-dimensional problem into n 1-dimensional subproblems each of which is optimized by a separated GA. Van den Bergh et al. [20] firstly proposed two improved PSO by integrating cooperative coevolution strategy into PSO, called CPSO-SK and CPSO-HK, which divides an n-dimensional problem into k s-dimensional subproblems. Similarly, Mohammed EI-Abd [21] presented two cooperative coevolutionary ABC algorithms, namely, CABC-S and CABC-H. Shi et al. [22] applied the cooperative coevolution strategy to the differential evolution algorithm, called the cooperative coevolutionary differential evolution (CCDE). The algorithm partitions high-dimensional solution space into two equal-sized subcomponents but it does not perform well as the dimensionality increases. For a fully separable problem with no interrelationship between variables, each variable can be optimized independently to obtain the optimal solution for the entire problem. However, for the nonseparable problem of the relationship between variables, the optimization effect of the fixed-scale grouping strategy will be worse, and even the optimal solution will not be obtained. Therefore, many scholars began to study the decomposition strategy that can solve the nonseparable problems.

The dynamic grouping-based CC methods dynamically adjust the size of the subpopulation to accommodate different types of large-scale optimization problems. They are efficient to handle nonseparable problems. Yang et al. [23] proposed
<table>
<thead>
<tr>
<th>Year</th>
<th>Author</th>
<th>Algorithm</th>
<th>Year</th>
<th>Author</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>Potter and De Jong [18]</td>
<td>CCGA-1, CCGA-2</td>
<td>2011</td>
<td>Omidvar et al. [29]</td>
<td>CRCC</td>
</tr>
<tr>
<td>1999</td>
<td>Weicker, K et al. [32]</td>
<td>CCLVI</td>
<td>2012</td>
<td>Sayed et al. [33]</td>
<td>HDIMA</td>
</tr>
<tr>
<td>2001</td>
<td>Liu et al. [34]</td>
<td>FEPCCC</td>
<td>2012</td>
<td>Sayed et al. [35]</td>
<td>DIMA</td>
</tr>
<tr>
<td>2005</td>
<td>Shi et al. [22]</td>
<td>CCDE</td>
<td>2013</td>
<td>Liu and Tang [37]</td>
<td>CC-CMA-ES</td>
</tr>
<tr>
<td>2008</td>
<td>Yang et al. [38]</td>
<td>MLCC</td>
<td>2014</td>
<td>Omidvar et al. [39]</td>
<td>MLSof</td>
</tr>
<tr>
<td>2009</td>
<td>Li et al. [24]</td>
<td>C CPSO</td>
<td>2014</td>
<td>Mahdavi et al. [40]</td>
<td>DM-HDMR</td>
</tr>
<tr>
<td>2009</td>
<td>Ray et al. [26]</td>
<td>CCEA-AVP</td>
<td>2015</td>
<td>Kazimipour et al. [41]</td>
<td>CBCC</td>
</tr>
<tr>
<td>2010</td>
<td>Chen et al. [27]</td>
<td>CCVIL</td>
<td>2016</td>
<td>Peng et al. [43]</td>
<td>mCCSEA</td>
</tr>
<tr>
<td>2010</td>
<td>Singh et al. [44]</td>
<td>CCEA-AVP</td>
<td>2017</td>
<td>Peng et al. [45]</td>
<td>MMO-CC</td>
</tr>
<tr>
<td>2011</td>
<td>Omidvar et al. [47]</td>
<td>DECC-D</td>
<td>2018</td>
<td>Peng et al. [48]</td>
<td>CC-SMP</td>
</tr>
</tbody>
</table>
a new cooperative coevolution framework that is capable of optimizing large-scale nonseparable problems. It uses a random grouping strategy to divide the problem solution space into multiple subspaces of different sizes. Li et al. [24] embedded a random grouping strategy and an adaptive weighting mechanism into the particle swarm optimization algorithm and proposed a cooperative coevolutionary particle swarm optimization (CCPSO) algorithm. The results of the random grouping method show the effective performance to solve the scalable nonseparable benchmark function (up to 1000D). However, as the number of interaction variables increases, its performance becomes invalid [25]. Therefore, some scholars consider adding a priori knowledge of the problem in the process of algorithm evolution and propose many learning-based dynamic grouping strategies. Ray et al. [26] discussed the effects of problem size, the number of subpopulations, and the number of iterations on some separable and nonseparable problems in cooperative coevolutionary algorithms. Then, a Cooperative Coevolutionary Algorithm with Correlation based Adaptive Variable Partitioning (CCEA-AVP) is proposed to deal with scalable nonseparable problems. Chen et al. [27] proposed a CC method with Variable Interaction Learning (CCVIL) which is able to adaptively change group sizes. Considering that there is no prior knowledge that cannot decompose the problem, Omidvar et al. [28] proposed an automatic decomposition strategy, which can keep the correlation between decision variables at the minimum.

2.2. Nondecomposition Strategy. The other is nondecomposition methods. Different from the decomposition-based strategy, the nondecomposition methods mainly study the algorithm itself and improve the corresponding operator in order to improve the performance of the algorithm for solving large-scale global optimization problems. Nondecomposition-based methods mainly include swarm intelligence, evolutionary computation, and local search-based approaches. Hsieh et al. [49] added variable particles and information sharing mechanism to the basic particle swarm optimization algorithm and proposed a variation called the Efficient Population Utilization Strategy for Particle Swarm Optimizer (EPUS-PSO). A modified PSO algorithm with a new velocity updating method, as the rotated particle swarm, was proposed in literature [50]. It transforms coordinates and uses information from other dimensions to maintain the diversity of each dimension. Fan et al. [51] proposed a novel particle swarm optimization approach with dynamic neighborhood that is based on kernel fuzzy clustering and variable trust region methods (called FT-DNPSO) for large-scale optimization. In terms of evolutionary computation, Hedar et al. [52] modified genetic algorithm with new strategies of population partitioning and space reduction for high-dimensional problems. In order to improve the performance of differential evolution algorithm, Zhang et al. [53] proposed an adaptive differential evolution algorithm (called JADE) using a new mutation and adaptive control parameter strategy. Local search-based approaches have been recognized as an effective algorithm framework for solving optimization problems. Hvattum et al. [54] introduced a new direct search method, based on Scatter Search, designed to remedy the lack of a good derivative-free method for solving problems of high dimensions. Liu et al. [55] improved the local search depth parameter in the memetic algorithm and proposed an adaptive local search depth (ALSD) strategy, which can arrange the local search computing resources according to its performance dynamically.

2.3. Related Work of GWO. Grey Wolf Optimizer (GWO) is a recently proposed swarm intelligence algorithm inspired by the social hierarchy and hunting behavior of wolves. It has the advantages of less control parameters, high solution accuracy, and fast convergence speed. Compared with other classical metaheuristic algorithms such as genetic algorithm (GA), particle swarm optimization (PSO), and gravitational search algorithm (GSA), the GWO shows powerful exploration capability and better convergence characteristics. Owing to its simplicity and ease of implementation, GWO has gained significant attention and has been applied in solving many practical optimization problems since its invention.

Recently, many scholars have applied the GWO to different optimization problems and also proposed many varieties in order to improve the performance of GWO. Saremi et al. [56] introduced a dynamic population updating mechanism in the GWO, removing individuals with poor fitness in the population and regenerating new individuals to replace them in order to enhance the exploration ability of GWO. However, the diversity of grey population was reduced by this strategy, which leads to the algorithm being local optimum. Zhu et al. [57] integrated the idea of differential evolution into the GWO, which prevented the early stagnation of the GWO algorithm and accelerated the convergence speed. However, it fails to converge in high-dimensional problems. Kumar et al. [58] designed a novel variant of GWO for numerical optimization and engineering design problems. They utilized the prey weight and astrophysics-based learning concept to enhance the exploration ability. However, this method also does not show good performance in solving high-dimensional function problems. In the application of the GWO algorithm, Guha et al. [59] applied the grey wolf algorithm to the load control problem in large-scale power systems. In literature [60], a multiobjective discrete grey wolf algorithm is proposed for the uniqueness of welding job scheduling problem. The results show that the algorithm can solve the scheduling problem in practical engineering. Emary et al. [61] proposed a new binary grey wolf optimization algorithm for the selection of the best feature subset.

The swarm intelligence optimization algorithm has natural parallelism, so it has advantages in solving large-scale optimization problems. In order to study the ability of grey wolf optimization algorithm to solve high-dimensional functions, Long et al. [62] proposed a nonlinear adjustment strategy for convergence factors, which balanced well the exploration and exploitation of the algorithm. But this strategy also does not solve the high-dimensional function optimization problem well. In 2017, Gupta et al. [63] studied the performance of the GWO algorithm for solving 100 to 1000-dimensional function optimization problems. The results indicate that
GWO is a powerful nature-inspired optimization algorithm for large-scale problems, except Rosenbrock function.

In summary, the decomposition-based CC strategy has a good effect in solving the nonseparable problem, but it cannot solve the imbalance problem well, and it still needs to be improved in the accuracy of the solution. In addition, in the method based on nondecomposition strategy, although the performance of many algorithms has been greatly improved, but the research on hybrid algorithms is still rare. Therefore, this paper combines three genetic operators of selection, crossover, and mutation with GWO algorithm and proposes a Hybrid Genetic Grey Wolf Algorithm, (HGGWA). The main improvement strategies of the algorithm are as follows: (1) initialize the population based on the Opposition-Based Learning strategy; (2) the nonlinear adjustment of parameters effectively balances exploration and exploitation capabilities while accelerating the convergence speed of the algorithm; (3) select the population through a combination of elite retention strategy and gambling roulette; (4) based on the method of dimensionality reduction and population division, the whole population is divided into multiple subpopulations for cross-operation to increase the diversity of the population; (5) perform mutation operations on elite individuals in the population to prevent the algorithm from falling into local optimum.

3. Grey Wolf Optimization Algorithm

3.1. Mathematical Models. In the GWO algorithm [12], the position of $\alpha$ wolf, $\beta$ wolf, and $\delta$ wolf is the fittest solution, the second-best solution, and the third-best solution, respectively. And the positions of the $\omega$ wolves are the remaining candidate solutions. In the whole process of searching for prey in the space, the $\omega$ wolves gradually update their positions according to the optimal position of the $\alpha$, $\beta$, and $\delta$ wolves. So they can gradually approach and capture the prey, that is, to complete the process of searching for the optimal solution. The position updating imaginary diagram in GWO is shown in Figure 1.

The grey wolves position update formula in this process is as follows:

$$
\vec{D} = \left| \vec{C} * \vec{X}_p(t) - \vec{X}(t) \right| \\
\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} * \vec{D}
$$

(1)

(2)

where $\vec{D}$ indicates the distance between the grey wolf individual and the prey, $t$ indicates the current iteration, $\vec{A}$ and $\vec{C}$ are the coefficient vectors, $\vec{A}$ is a convergence coefficient used to balance the exploration and the exploitation, $\vec{C}$ is used to simulate the effects of nature, $\vec{X}_p$ is the position vector of the prey, and $\vec{X}$ indicates the position vector of a grey wolf.

The coefficient vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$
\vec{A} = 2 \vec{a} \ast r_1 - \vec{a} \\
\vec{C} = 2 \ast r_2
$$

(3)

(4)

where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations and $r_1$ and $r_2$ are random vectors in $[0,1]$. At the same time, the fluctuation range of the coefficient vector $\vec{A}$ gradually decreases as $\vec{a}$ decreases.

Thus, according to (1) and (2), the grey wolf individual randomly moves within the search space to gradually approach the optimal solution. The specific location update formula is as follows:

$$
\vec{D}_\alpha = \left| \vec{C}_1 * \vec{X}_\alpha - \vec{X} \right|, \\
\vec{D}_\beta = \left| \vec{C}_2 * \vec{X}_\beta - \vec{X} \right|, \\
\vec{D}_\delta = \left| \vec{C}_3 * \vec{X}_\delta - \vec{X} \right|, \\
\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 * \vec{D}_\alpha, \\
\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 * \vec{D}_\beta, \\
\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 * \vec{D}_\delta, \\
\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}
$$

(5)

(6)

(7)

(8)

(9)

(10)

(11)

where $\vec{D}_\alpha, \vec{D}_\beta, \vec{D}_\delta$ indicate the distance between the $\alpha, \beta, \delta$ wolves and the prey, respectively. $\vec{X}_1, \vec{X}_2, \vec{X}_3$ represent the positional components of the remaining grey wolves based on the positions of $\alpha, \beta, \delta$ wolves, respectively. $\vec{X}(t+1)$ represents the position vector after the grey wolf is updated.

3.2. Basic Process of GWO.

Step 1. Randomly generate an initial population, and initialize parameters $\vec{a}$, $\vec{A}$, and $\vec{C}$.

Step 2. Calculate the fitness of the grey wolf individual, and save the positions of the first three individuals with the highest fitness as $\vec{X}_\alpha, \vec{X}_\beta,$ and $\vec{X}_\delta$.

Step 3. Update the position information of each grey wolf according to (5) to (11), whereby obtaining the next generation population, and then update the values of parameters $\vec{a}$, $\vec{A}$, and $\vec{C}$.

Step 4. Calculate the fitness of individuals in the new population, and update $\vec{X}_\alpha, \vec{X}_\beta, \text{and} \vec{X}_\delta$.

Step 5. Repeat Steps 2–4 until the optimal solution is obtained or the maximum number of iterations is reached.

4. Hybrid Genetic Grey Wolf Algorithm

This section describes the details of HGGWA, the hybrid algorithm proposed in this paper. Firstly, the initial population strategy based on Opposition-Based Learning is
described in detail in Section 4.1. Secondly, Section 4.2 introduces the nonlinear adjustment strategy for parameter $a$. Then three operations of selection, crossover, and mutation are carried out in Sections 4.3–4.5, respectively. Finally, the time complexity of HGGWA is analyzed in Section 4.6.

4.1. Initial Population Strategy Based on Opposition-Based Learning. For the swarm intelligence optimization algorithm based on population iteration, the diversity of initial populations lays the foundation for the efficiency of the algorithm. The better diversity of the population will reduce the computation time and improve the global convergence of the algorithm [64]. However, like other algorithms, the GWO algorithm also uses random initialization when generating populations, which will have a certain impact on the search efficiency of the algorithm. Opposition-Based Learning is a strategy proposed by Tizhoosh [65] to improve the efficiency of algorithm search. It uses the opposite points of known individual positions to generate new individual positions, thereby increasing the diversity of search populations. Currently, the Opposition-Based Learning strategy has been successfully applied to multiple swarm optimization algorithms [66, 67]. The opposite point is defined as follows.

Assume that there is an individual $X(x_1, x_2, \ldots, x_D)$ in the population $P$, then the opposite point of the element $x_i$ ($i = 1, 2, \ldots, D$) on each dimension is $x'_i = \ell_i + u_i - x_i$, where $\ell_i$ and $u_i$ are the lower bound and the upper of the $i$th dimension, respectively. According to the above definition, the Opposition-Based Learning strategy initializes the population as follows:

(a) Randomly initialize the population $P$, calculate the opposite point $x'_i$ of each individual position $x_i$, and all the opposite points constitute the opposition population $P'$.

(b) Calculate the fitness of each individual in the initial population $P$ and the opposition population $P'$, and arrange them in descending order.

(c) Select the top $N$ grey wolf individuals with the highest fitness as the final initial population.

4.2. Parameters Nonlinear Adjustment Strategy. It is significantly important how to balance the exploration and exploitation for swarm intelligence algorithms. In the early iteration of the algorithm, the powerful exploration capability is beneficial for the algorithm to expand the search range, so that the algorithm can search for the optimal solution with greater probability. In the later stage of the algorithm’s iteration, the effective exploitation ability can speed up the optimization process of the algorithm and improve the convergence accuracy of the final solution. Therefore, only when the swarm optimization algorithm can better coordinate its global exploration and local exploitation capabilities can it have strong robustness and faster convergence speed.

It can be seen from (2) and (3) that the dynamic change of the parameter $a$ plays a crucial role in the algorithm. In the basic GWO, the linear decrement strategy of parameter $a$ does not reflect the actual convergence process of the algorithm. Therefore, the algorithm does not balance the exploration and exploitation capabilities well, especially when solving large-scale multimodal function problems. In order to dynamically adjust the global exploration and local exploitation process of the algorithm, this paper proposes a nonlinear adjustment strategy for the parameter $a$, which is beneficial for HGGWA to search for the optimal solution. The improved parameter nonlinear adjustment equation is as follows:

$$a = a_{\text{max}} + (a_{\text{min}} - a_{\text{max}}) \cdot \left(\frac{1}{1 + e^{t/M_{\text{iter}}}}\right)^k$$

(12)

where $a_{\text{max}}$ and $a_{\text{min}}$ are the initial value and the end value of the parameter $a$, $t$ is the current iteration index, $M_{\text{iter}}$ is the maximum number of iterations, and $k$ is nonlinear adjustment coefficient.

Thus, the variation range of the convergence factor $A$ is controlled by the nonlinear adjustment of the parameter $a$. When $|A| > 1$, the grey wolf population expands the search range to find better prey, which corresponds to the global exploration of the algorithm; when $|A| < 1$,
the whole population shrinks the search range; thus, an encirclement is formed around the prey to complete the final attacking against the prey, which corresponds to the local exploitation process of the algorithm. The whole process has a positive effect on balancing global search and local search, which is beneficial to improve the accuracy of the solution and further accelerate the convergence speed of the algorithm.

4.3. Selection Operation Based on Optimal Retention. For the intelligent optimization algorithm based on population evolution, the evolution of each generation of population directly affects the optimization effect of the algorithm. In order to inherit the superior individuals in the paternal population to the next generation without being destroyed, it is necessary to preserve the good individuals directly. The optimal retention selection strategy is an effective way to preserve good individuals in genetic algorithms. This paper integrates them into the GWO to improve the efficiency of the algorithm. Assuming that the current population is \(P(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_D)\), the fitness of the individual \(\mathbf{X}_i\) is \(F_i\). The specific operation is as follows:

(a) Calculate the fitness \(F_i\) \((i = 1, 2, \ldots, D)\) of each individual and arrange them in descending order.

(b) Selecting the individuals with the highest fitness to copy directly into the next generation of population.

(c) Calculate the total fitness \(S\) of the remaining individuals and the probability \(P_i\) that each individual is selected.

\[
S = \sum_{i=1}^{D-1} F_i \quad (i = 1, 2, \ldots, D - 1) \tag{13}
\]

\[
P_i = \frac{F_i}{\sum_{i=1}^{D-1} F_i} \quad (i = 1, 2, \ldots, D - 1) \tag{14}
\]

(d) Calculate the cumulative fitness value \(s_i\) for each individual, and then the selection operation is performed in the manner of the bet roulette until the number of individuals in the children population is consistent with the parent population.

\[
s_i = \frac{\sum_{i=1}^{D-1} F_i}{S} \quad (i = 1, 2, \ldots, D - 1) \tag{15}
\]

4.4. Crossover Operation Based on Population Partitioning Mechanism. Due to the decrease of population diversity in the late evolution, the GWO algorithm is easy to fall into local optimum for solving the large-scale high-dimensional optimization problem, in order to overcome the problems caused by large-scale and complexity and to ensure that the algorithm searches for all solution spaces.

In the HGGWA algorithm, the whole population \(P\) is divided into \(v \times \eta\) subpopulations \(P_{i,j}\) \((i = 1, \ldots, v; j = 1, \ldots, \eta)\). The optimal size of each subpopulation \(P_{i,j}\) was tested to be \(5 \times 5\). Individuals of each subpopulation were cross-operated to increase the diversity of the population. The specific division method is shown below.

The initial population: \(P\)

\[
\begin{bmatrix}
\mathbf{X}_{1,1} & \mathbf{X}_{1,2} & \cdots & \mathbf{X}_{1,d} \\
\mathbf{X}_{2,1} & \mathbf{X}_{2,2} & \cdots & \mathbf{X}_{2,d} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{X}_{P,1} & \mathbf{X}_{P,2} & \cdots & \mathbf{X}_{P,d}
\end{bmatrix}
\]

Subpopulations: \(P_{i,j}\)

\[
\begin{bmatrix}
\mathbf{X}_{1,1} & \cdots & \mathbf{X}_{1,\eta} \\
\vdots & \ddots & \vdots \\
\mathbf{X}_{5,1} & \cdots & \mathbf{X}_{5,\eta}
\end{bmatrix}, \begin{bmatrix}
\mathbf{X}_{5,(d-4)} & \cdots & \mathbf{X}_{5,d} \\
\vdots & \ddots & \vdots \\
\mathbf{X}_{P,(d-4)} & \cdots & \mathbf{X}_{P,d}
\end{bmatrix}
\]

In genetic algorithms, cross-operation has a very important role and is the main way to generate new individuals. In the improved algorithm of this paper, each individual in the subpopulation is cross-operated in a linear crossover manner. Generating a corresponding random number \(r_i \in (0, 1)\) for each individual \(\mathbf{X}_i\) in the subpopulation. When the random number \(r_i\) is less than the crossover probability \(P_c\), the corresponding individual \(\mathbf{X}_i\) is paired for cross-operation.

An example is as follows: Crossover (p1, p2)

(a) Generate a random number \(\lambda \in (0, 1)\).

(b) Two children \(c^1(c_1^1, \ldots, c_1^2), c^2(c_2^1, \ldots, c_2^2)\) are generated by two parents \(p^1(p_1^1, \ldots, p_1^2)\) and \(p^2(p_2^1, \ldots, p_2^2)\):

\[
c_1^i = \lambda p_1^i + (1 - \lambda) p_2^i, \quad i = 1, 2, \ldots, D \tag{17}
\]

\[
c_2^i = \lambda p_2^i + (1 - \lambda) p_1^i, \quad i = 1, 2, \ldots, D \tag{18}
\]

4.5. Mutation Operation for Elite Individuals. Due to the existence of the selection operation based on the optimal preservation strategy, the grey wolf individuals in the whole population are concentrated in a small optimal region in the later stage of the iterative process, which easily leads to the loss of population diversity. If the current optimal individual is a locally optimal solution, then the algorithm is easy to fall into local optimum, especially when solving high-dimensional multimodal functions. To this end, this paper introduces the mutation operator in the HGGWA algorithm to perform mutation operations on elite individuals in the population. The specific operations are as follows.

Assume that the optimal individual is \(\mathbf{x}_k(x_1, x_2, \ldots, x_D)\) and the mutation operation is performed on \(x_i\) with the mutation probability \(P_m\). That is, select a gene \(x_k\) from the optimal individual with probability \(P_m\), instead of the gene \(x_k\) with a random number between upper and lower bounds to
generate a new individual \( x'_i = (x'_1, x'_2, \ldots, x'_d) \). The specific operation is as follows:

\[
x'_i = \begin{cases} 
  l + \lambda \cdot (u - l) & i = k \\
  x_i & i \neq k 
\end{cases}
\]

where \( \lambda \) is a random number in \([0, 1]\) and \( l \) and \( u \) are the lower and upper bounds of the individual \( x_i \), respectively.

4.6. Pseudo Code of the HGGWA and Time Complexity Analysis. This section describes how to calculate an upper bound for the total number of fitness evaluations (FE) required by HGGWA. As shown in Algorithm 1, the computational complexity of HGGWA in one generation is mainly dominated by the position updating of all search agents in line (10). The position updating requires a time complexity of \( O(N \cdot dim \cdot Max_{iter}) \) (\( N \) is the population size, \( dim \) is the dimensions of each benchmark function, and \( Max_{iter} \) is the maximum number of iterations of HGGWA) to obtain the needed parameters and to finish the whole position updating progress of all search agents. In addition, the process of crossover operations is another time-consuming step in line (20). It needs a time complexity of \( O(2 \cdot v \cdot \eta \cdot Max_{iter}) \) (\( v = N/5, \eta = \text{dim}/5 \)) to complete the crossing of individuals in each subpopulation according to the crossover probability \( P_c \). It should be noted that this is only the worst case computational complexity. If there are only two individuals in each subpopulation that need to be crossed, then the minimum amount of computation is \( O(\text{dim} \cdot \text{Max}_{iter}) \).

Therefore, the maximum computational complexity caused by the two dominant processes in the algorithm is \( \max \{ O(N \cdot dim \cdot Max_{iter}), O(2 \cdot v \cdot \eta \cdot Max_{iter}) \} \) (\( v = N/5, \eta = \text{dim}/5 \)) in one generation, i.e., \( O(N \cdot \text{dim} \cdot \text{Max}_{iter}) \).

5. Numerical Experiments and Analysis

In this section, the proposed HGGWA algorithm will be evaluated on both classical benchmark functions [68] and the suite of test functions provided by CEC2008 Special Session on large-scale global optimization. The algorithms used for comparison include not only conventional EAs but also other CC optimization algorithms. Experimental results are provided to analyze the performance of HGGWA in the context of large-scale optimization problems. In addition, the sensitivity analysis of nonlinear adjustment coefficient and the global convergence of HGGWA are discussed in Sections 5.4 and 5.5, respectively.

5.1. Benchmark Functions and Performance Measures. In order to verify the ability of HGGWA algorithm to solve high-dimensional complex functions, 10 general high-dimensional benchmark functions (100D, 500D, 1000D) were selected for optimization test. Among them, \( f_1 \sim f_6 \) are unimodal functions, which are used to test the local searchability of the algorithm; \( f_7 \sim f_{10} \) are multimodal functions, which are used to test the global searchability of the algorithm. These test functions have multiple local optima points with uneven distribution, nonconvexity, and strong oscillation, which are very difficult to converge to the global optimal solution, especially in the case of being high-dimensional. The specific
characteristics of the 10 benchmark functions are shown in Table 2.

In order to show the optimization effect of the algorithm more clearly, this paper selects two indicators to evaluate the performance of the algorithm [69]. The first is the solution accuracy (AC), which reflects the difference between the optimal results of the algorithm and the theoretical optimal value. In an experiment, if the final convergence result of the algorithm is less than the AC, then the optimization is considered successful. Assuming that the optimal value of a certain optimization is \( X_{\text{opt}} \), and the theoretical optimal value is \( X_{\text{min}} \), then the AC is calculated as follows:

\[
AC = \left| X_{\text{opt}} - X_{\text{min}} \right|
\]  

The second is the successful ratio (SR), which reflects the proportion of the number of successful optimizations to the total number of optimizations under the condition that the algorithm has a certain accuracy. Assume that in the \( T \) iterations of the algorithm, the number of successful optimizations is \( t \), then the SR is calculated as follows:

\[
SR = \frac{t}{T} \times 100\%
\]

5.2. Results on Classical Benchmark Functions. In this section, a set of simulation tests were performed in order to verify the effectiveness of HGGWA. Firstly, the 10 high-dimensional benchmark functions in Table 2 are optimized by HGGWA algorithm proposed in this paper, and the results are compared with WOA, SSA, and ALO. In addition, the running time of several algorithms is compared under the given convergence accuracy.

5.2.1. Parameters Settings for the Compared Algorithms. In all experiments, the values of the common parameters, such as the maximum number of iterations (MaxIter), the dimension of the functions (D), and the population sizes (N) were chosen the same. For all test problems, we focus on investigating the optimization performance of the proposed method on problems with \( D = 100, 500, \) and \( 1000 \). The maximum number of iterations is set to 1000, and the population size is set to 50. The parameter setting of WOA, SSA, and ALO is derived from the original literature [70–72]. The optimal parameter settings of the HGGWA algorithm proposed in this paper are shown in Table 3.

For each experiment of an algorithm on a benchmark function, 30 independent runs are performed to obtain a
5.2.2. Comparison with State-of-the-Art Metaheuristic Algorithms. In order to verify the efficiency of the HGGWA algorithm, several state-of-the-art metaheuristic algorithms recently proposed are compared with their optimization results. These algorithms are Whale Optimization Algorithm (WOA), Salp Swarm Algorithm (SSA), and Ant Lion Optimization Algorithm (ALO). The test was performed using the same 10 high-dimensional functions in Table 2, and the results of the comparison are shown in Table 4.

From Table 4, it can be known that the convergence accuracy and optimization successful ratio of the HGGWA algorithm are significantly higher than the other three algorithms for most of the test functions. In the case of 100 dimensions, HGGWA algorithm can converge to the global optimal solution for the other nine test functions at one time in addition to function 3. And the algorithm has a successful ratio of 100% for the nine functions under the condition of certain convergence accuracy. The convergence result of the WOA algorithm is better than the other three algorithms for functions \(f_1\) and \(f_2\), and its robustness is excellent. With the increase of function dimension, the convergence precision of several algorithms decreases slightly, but the optimization results of HGGWA algorithm are better than those of WOA, SSA, and ALO. In general, the HGGWA algorithm exhibits better optimization effects than the WOA, SSA, and ALO algorithms in the 100-, 500-, and 1000-dimensional test functions, which proves the effectiveness of the HGGWA algorithm for solving high-dimensional complex functions.

Generally speaking, the HGGWA algorithm performs better than other algorithms in most test functions. In order to compare the convergence performance of the four algorithms more clearly, the convergence curves of the four algorithms on Sphere, Schwefel’s 2.22, Rastrigin, Griewank, Quartic, and Ackley functions with \(D=100\) are shown in Figure 2. Sphere and Schwefel’s 2.22 represent the two most basic unimodal functions. Rastrigin and Griewank represent the two most basic multimodal functions. It is seen that HGGWA has higher precision and faster convergence speed than other algorithms.

Table 3: Parameter settings of the HGGWA.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N)</td>
<td>Population size</td>
<td>50</td>
</tr>
<tr>
<td>(P_c)</td>
<td>Cross probability</td>
<td>0.8</td>
</tr>
<tr>
<td>(P_m)</td>
<td>Mutation probability</td>
<td>0.01</td>
</tr>
<tr>
<td>(Max_{iter})</td>
<td>Maximum number of iterations</td>
<td>1000</td>
</tr>
<tr>
<td>(k)</td>
<td>Nonlinear adjustment coefficient</td>
<td>0.5</td>
</tr>
</tbody>
</table>

5.2.3. Comparative Analysis of Running Time. To evaluate the actual runtime of the several compared algorithms, including SSA, WOA, GWO, and HGGWA, their average running times (in seconds: s) from 30 runs on function \(f_1, f_2, f_6, f_7, f_8, f_9\) and \(f_{10}\) are plotted in Figure 3. The convergence accuracy of each test function has been described in Table 2.

Figure 3 shows the convergence behavior of different algorithms. Each point on the plot was calculated by taking the average of 30 independent runs. As can be seen from Figure 3, several algorithms show different effects under the same calculation accuracy. For the functions \(f_1, f_2, f_7, f_8, f_9\), the GWO algorithm has the fastest convergence rate as a result of the advantages of the GWO algorithm itself. However, for the test functions \(f_6\) and \(f_{10}\), the running time of the HGGWA algorithm is the shortest. Through further analysis, it is known that HGGWA algorithm adds a little computing time compared to GWO algorithm because of several improved strategies such as selection, crossover, and mutation. However, the convergence speed of HGGWA algorithm is still better than SSA and WOA algorithms. Overall, the convergence speed of the HGGWA algorithm performs better in several algorithms compared. Moreover, the running time of different test functions is very small, and it is kept between 1s and 3s, which shows that the HGGWA algorithm has excellent stability.

5.3. Results on CEC2008 Benchmark Functions. This section provides an analysis of the effectiveness of HGGWA in terms of the CEC2008 large-scale benchmark functions and a comparison with four powerful algorithms which have been proven to be effective in solving large-scale optimization problems. Experimental results are provided to analyze the performance of HGGWA for large-scale optimization problems.

We tested our proposed algorithms with CEC2008 functions for 100, 500, and 1000 dimensions and the means of the best fitness value over 50 runs were recorded. While \(F1\) (Shifted Sphere), \(F4\) (Shifted Rastrigin), and \(F6\) (Shifted Ackley) are separable functions, \(F2\) (Schwefel Problem), \(F3\) (Shifted Rosenbrock), \(F5\) (Shifted Griewank), and \(F7\) (Fast Fractal) are nonseparable, presenting a greater challenge to any algorithm that is prone to variable interactions. The performance of HGGWA is compared with other algorithms CCPSO2 [31], DEwSAcc [5], MLCC [38], and EPUS-PSO [49] for 100, 500, and 1000 dimensions, respectively. The maximum number of fitness evaluations (FEs) was calculated by the following formula, \(FEs = 5000 \times D\), where \(D\) is the number of dimensions. In order to reflect the fairness of the comparison results, the optimization results of the other four algorithms are directly derived from the original literature. The specific comparison results are shown in Table 5.

Table 5 shows the experimental results and the entries shown in bold are significantly better than other algorithms. A general trend that can be seen is that HGGWA algorithm has a promising optimization effect for most of the 7 functions.

Generally speaking, HGGWA outperforms CCPSO2 on 6 out of 7 functions with 100 and 500 dimensions, respectively. Among them, the result of HGGWA is slightly worse than...
<table>
<thead>
<tr>
<th>Function</th>
<th>Dim</th>
<th>Mean</th>
<th>Std</th>
<th>SR</th>
<th>Mean</th>
<th>Std</th>
<th>SR</th>
<th>Mean</th>
<th>Std</th>
<th>SR</th>
<th>Mean</th>
<th>Std</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WOA</td>
<td>SSA</td>
<td>ALO</td>
<td>HGGWA</td>
<td>WOA</td>
<td>SSA</td>
<td>ALO</td>
<td>HGGWA</td>
<td>WOA</td>
<td>SSA</td>
<td>ALO</td>
<td>HGGWA</td>
<td></td>
</tr>
<tr>
<td>$f_1$</td>
<td>100</td>
<td>5.77E-96</td>
<td>4.69E-96</td>
<td>100%</td>
<td>1.01E-02</td>
<td>1.24E-02</td>
<td>0</td>
<td>3.04E-01</td>
<td>2.31E-01</td>
<td>0</td>
<td>4.85E-53</td>
<td>2.41E-52</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>7.24E-93</td>
<td>2.98E-94</td>
<td>100%</td>
<td>3.24E+04</td>
<td>2.18E+04</td>
<td>0</td>
<td>7.34E+04</td>
<td>1.32E+04</td>
<td>0</td>
<td>8.35E-25</td>
<td>8.86E-25</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>2.05E-89</td>
<td>3.16E-90</td>
<td>100%</td>
<td>1.19E+05</td>
<td>1.04E+05</td>
<td>0</td>
<td>2.48E+05</td>
<td>2.12E+05</td>
<td>0</td>
<td>1.75E-17</td>
<td>2.73E-17</td>
<td>100%</td>
</tr>
<tr>
<td>$f_2$</td>
<td>100</td>
<td>1.89E-99</td>
<td>2.33E-99</td>
<td>100%</td>
<td>8.85E+00</td>
<td>2.37E+00</td>
<td>0</td>
<td>4.29E+02</td>
<td>3.04E+01</td>
<td>0</td>
<td>9.58E-18</td>
<td>2.36E-18</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>1.38E-99</td>
<td>2.16E-99</td>
<td>100%</td>
<td>3.37E+02</td>
<td>2.17E+02</td>
<td>0</td>
<td>2.31E+03</td>
<td>2.03E+03</td>
<td>0</td>
<td>9.58E-18</td>
<td>2.36E-18</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1.38E-99</td>
<td>2.16E-99</td>
<td>100%</td>
<td>8.45E+02</td>
<td>3.43E+02</td>
<td>0</td>
<td>3.25E+04</td>
<td>4.13E+04</td>
<td>0</td>
<td>2.50E-11</td>
<td>1.66E-11</td>
<td>100%</td>
</tr>
<tr>
<td>$f_3$</td>
<td>100</td>
<td>3.84E+01</td>
<td>2.17E+01</td>
<td>0</td>
<td>2.22E+01</td>
<td>1.05E+01</td>
<td>0</td>
<td>2.47E+01</td>
<td>1.34E+01</td>
<td>0</td>
<td>4.55E+01</td>
<td>3.69E+01</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>9.32E+01</td>
<td>5.32E+01</td>
<td>0</td>
<td>3.27E+01</td>
<td>2.33E+01</td>
<td>0</td>
<td>3.92E+01</td>
<td>2.34E+01</td>
<td>0</td>
<td>7.22E+01</td>
<td>5.01E+00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>9.86E+01</td>
<td>3.24E+01</td>
<td>0</td>
<td>3.31E+01</td>
<td>1.34E+01</td>
<td>0</td>
<td>4.93E+01</td>
<td>3.99E+01</td>
<td>0</td>
<td>8.31E+01</td>
<td>4.36E+00</td>
<td>0</td>
</tr>
<tr>
<td>$f_4$</td>
<td>100</td>
<td>9.67E+01</td>
<td>3.98E+01</td>
<td>0</td>
<td>2.22E+01</td>
<td>1.05E+01</td>
<td>0</td>
<td>2.47E+01</td>
<td>1.34E+01</td>
<td>0</td>
<td>4.55E+01</td>
<td>3.69E+01</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>4.95E+02</td>
<td>3.14E+01</td>
<td>0</td>
<td>3.27E+01</td>
<td>2.33E+01</td>
<td>0</td>
<td>3.92E+01</td>
<td>2.34E+01</td>
<td>0</td>
<td>7.22E+01</td>
<td>5.01E+00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>9.91E+02</td>
<td>2.48E+02</td>
<td>0</td>
<td>3.31E+01</td>
<td>1.34E+01</td>
<td>0</td>
<td>4.93E+01</td>
<td>3.99E+01</td>
<td>0</td>
<td>8.31E+01</td>
<td>4.36E+00</td>
<td>0</td>
</tr>
<tr>
<td>$f_5$</td>
<td>100</td>
<td>7.83E+05</td>
<td>2.36E+05</td>
<td>0</td>
<td>2.22E+01</td>
<td>1.05E+01</td>
<td>0</td>
<td>2.47E+01</td>
<td>1.34E+01</td>
<td>0</td>
<td>4.55E+01</td>
<td>3.69E+01</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>1.98E+07</td>
<td>3.44E+07</td>
<td>0</td>
<td>3.27E+01</td>
<td>2.33E+01</td>
<td>0</td>
<td>3.92E+01</td>
<td>2.34E+01</td>
<td>0</td>
<td>7.22E+01</td>
<td>5.01E+00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>7.83E+05</td>
<td>2.36E+05</td>
<td>0</td>
<td>2.22E+01</td>
<td>1.05E+01</td>
<td>0</td>
<td>2.47E+01</td>
<td>1.34E+01</td>
<td>0</td>
<td>4.55E+01</td>
<td>3.69E+01</td>
<td>0</td>
</tr>
<tr>
<td>$f_6$</td>
<td>100</td>
<td>2.65E+04</td>
<td>1.43E+04</td>
<td>0</td>
<td>1.12E+00</td>
<td>1.03E+00</td>
<td>0</td>
<td>8.28E+01</td>
<td>2.68E+01</td>
<td>0</td>
<td>3.45E+05</td>
<td>2.77E-05</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>3.94E+04</td>
<td>3.22E+04</td>
<td>0</td>
<td>4.29E+05</td>
<td>3.16E+05</td>
<td>0</td>
<td>1.18E+06</td>
<td>1.03E+06</td>
<td>0</td>
<td>2.67E+03</td>
<td>4.38E+03</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>6.68E+07</td>
<td>3.44E+07</td>
<td>0</td>
<td>2.48E+06</td>
<td>1.23E+06</td>
<td>0</td>
<td>3.72E+06</td>
<td>3.49E+06</td>
<td>0</td>
<td>8.33E+00</td>
<td>5.12E+00</td>
<td>0</td>
</tr>
<tr>
<td>$f_7$</td>
<td>100</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>100%</td>
<td>1.52E-01</td>
<td>2.44E-01</td>
<td>0</td>
<td>1.74E+01</td>
<td>1.33E+01</td>
<td>0</td>
<td>2.73E-07</td>
<td>3.11E-07</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>100%</td>
<td>2.82E+02</td>
<td>2.13E+02</td>
<td>0</td>
<td>3.92E+02</td>
<td>1.08E+02</td>
<td>0</td>
<td>1.84E-16</td>
<td>8.44E-17</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>100%</td>
<td>1.10E+03</td>
<td>1.34E+03</td>
<td>0</td>
<td>3.61E+03</td>
<td>1.17E+03</td>
<td>0</td>
<td>2.31E-13</td>
<td>3.29E-14</td>
<td>100%</td>
</tr>
<tr>
<td>$f_8$</td>
<td>100</td>
<td>8.16E-03</td>
<td>4.32E-03</td>
<td>0</td>
<td>1.89E+01</td>
<td>2.19E+01</td>
<td>0</td>
<td>1.74E+01</td>
<td>1.33E+01</td>
<td>0</td>
<td>2.73E-07</td>
<td>3.11E-07</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>2.52E-02</td>
<td>1.99E-02</td>
<td>0</td>
<td>5.16E+01</td>
<td>3.33E+01</td>
<td>0</td>
<td>3.97E+05</td>
<td>2.96E+05</td>
<td>0</td>
<td>8.12E-05</td>
<td>4.98E-06</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1.43E-02</td>
<td>1.03E-02</td>
<td>0</td>
<td>1.00E+05</td>
<td>2.13E+04</td>
<td>0</td>
<td>6.96E+06</td>
<td>1.34E+06</td>
<td>0</td>
<td>6.33E-04</td>
<td>3.77E-03</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 4: Comparison results of 10 high-dimensional benchmark functions by WOA, SSA, ALO, and HGGWA.
Figure 2: Convergence curves of WOA, SSA, ALO, and HGGWA algorithms.

Figure 3: The average running times and iterations of SSA, WOA, GWO, and HGGWA on different test functions.
Table 5: Comparison results of 7 CEC’2008 benchmark functions by five algorithms.

<table>
<thead>
<tr>
<th>Function</th>
<th>Dim</th>
<th>CCPSO2 Mean</th>
<th>CCPSO2 Std</th>
<th>DEwSAcc Mean</th>
<th>DEwSAcc Std</th>
<th>MLCC Mean</th>
<th>MLCC Std</th>
<th>EPUS-PSO Mean</th>
<th>EPUS-PSO Std</th>
<th>HGGWA Mean</th>
<th>HGGWA Std</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>500</td>
<td>1000</td>
<td>100</td>
<td>500</td>
<td>100</td>
<td>500</td>
<td>100</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>F1</td>
<td></td>
<td>7.73E-14</td>
<td>3.00E-13</td>
<td>5.18E-13</td>
<td>6.08E+00</td>
<td>3.79E+01</td>
<td>7.82E+01</td>
<td>1.99E+01</td>
<td>3.33E+03</td>
<td>1.31E+01</td>
<td>1.38E+01</td>
</tr>
<tr>
<td>F2</td>
<td></td>
<td>1.37E-01</td>
<td>2.09E-09</td>
<td>9.61E-14</td>
<td>7.83E+00</td>
<td>4.21E+01</td>
<td>4.25E+01</td>
<td>9.60E+01</td>
<td>2.63E+02</td>
<td>5.18E+00</td>
<td>5.26E+00</td>
</tr>
<tr>
<td>F3</td>
<td></td>
<td>1.45E-02</td>
<td>1.54E+02</td>
<td>1.33E+03</td>
<td>8.65E+02</td>
<td>1.99E-01</td>
<td>4.06E-01</td>
<td>1.82E-01</td>
<td>2.63E+02</td>
<td>1.33E+03</td>
<td>1.38E+03</td>
</tr>
<tr>
<td>F4</td>
<td></td>
<td>3.07E-14</td>
<td>3.64E+02</td>
<td>3.98E-02</td>
<td>4.88E-03</td>
<td>1.99E-01</td>
<td>4.06E-01</td>
<td>1.82E-01</td>
<td>2.63E+02</td>
<td>1.33E+03</td>
<td>1.38E+03</td>
</tr>
<tr>
<td>F5</td>
<td></td>
<td>3.07E-14</td>
<td>3.64E+02</td>
<td>3.98E-02</td>
<td>4.88E-03</td>
<td>1.99E-01</td>
<td>4.06E-01</td>
<td>1.82E-01</td>
<td>2.63E+02</td>
<td>1.33E+03</td>
<td>1.38E+03</td>
</tr>
<tr>
<td>F6</td>
<td></td>
<td>3.07E-14</td>
<td>3.64E+02</td>
<td>3.98E-02</td>
<td>4.88E-03</td>
<td>1.99E-01</td>
<td>4.06E-01</td>
<td>1.82E-01</td>
<td>2.63E+02</td>
<td>1.33E+03</td>
<td>1.38E+03</td>
</tr>
<tr>
<td>F7</td>
<td></td>
<td>3.07E-14</td>
<td>3.64E+02</td>
<td>3.98E-02</td>
<td>4.88E-03</td>
<td>1.99E-01</td>
<td>4.06E-01</td>
<td>1.82E-01</td>
<td>2.63E+02</td>
<td>1.33E+03</td>
<td>1.38E+03</td>
</tr>
</tbody>
</table>
CCPSO2 for function F7 in the case of 100-dimensional; the two algorithms obtain similar optimization results for the function F6 in the case of 500-dimensional. In addition, at first glance it might seem that HGGWA and MLCC have similar performance. However, the HGGWA achieves better convergence accuracy than MLCC under the same number of iterations. In particular, the HGGWA algorithm’s optimization results are superior to DEwSAcc and EPUS-PSO in all dimensions. The result of HGGWA scaled very well from 100-D to 1000-D on F1, F4, F5, and F6, outperforming DEwSAcc and EPUS-PSO on all the cases.

5.4. Sensitivity Analysis of Nonlinear Adjustment Coefficient $k$.
This section mainly discusses the effect of nonlinear adjustment coefficients on the convergence performance of the algorithm. In the HGGWA algorithm, the role of parameter $a$ is to balance global exploration capabilities and local exploitation capabilities. In (12), the nonlinear adjustment coefficient $k$ is a key parameter and is mainly used to control the range of variation of the convergence factor. Therefore, we selected four different values for numerical experiments to analyze the effect of nonlinear adjustment coefficients on the performance of the algorithm. The four different values are 0.5, 1, 1.5, 2, and the results of the comparison are shown in Table 6. Among them, black bold represents the best result of solving in several algorithms.

In general, the value of the nonlinear adjustment coefficient $k$ has no significant effect on the performance of the HGGWA algorithm. On closer inspection, one can see that besides the function $f_3$, the optimization performance of the algorithm is optimal when the nonlinear adjustment coefficient $k$ is 0.5. But for function $f_5$, $k = 1.5$ is the best choice. For functions $f_7$ and $f_8$, the HGGWA algorithm obtains the optimal solution 0 in the case of several different values of the coefficient $k$. Therefore, for most test functions, the optimal value of the nonlinear adjustment factor $k$ is 0.5.

5.5. Global Convergence Analysis of HGGWA. In the HGGWA algorithm, the following four improved strategies are used to ensure the global convergence of the algorithm: (1) initial population strategy based on Opposition-Based Learning; (2) selection operation based on optimal retention; (3) crossover operation based on population partitioning mechanism; and (4) mutation operation for elite individuals.

The idea of solving the high-dimensional function optimization problem by HGGWA algorithm is as follows. Firstly, for the initial population strategy, the basic GWO generates the initial population in a random manner. However, it will have a great impact on the search efficiency of the algorithm for high-dimensional function optimization problems. The algorithm cannot effectively search the entire problem solution space if the initial grey wolf population is clustered in a small range. Therefore, the Opposition-Based Learning strategy is used to initialize the population, which can make the initial population evenly distributed in the solution space, thus laying a good foundation for the global search of the algorithm. Secondly, the optimal retention strategy is used to preserve the optimal individual of the parent population in the process of population evolution. As a result, the next generation of population can evolve in the optimal direction. In addition, the individuals with higher fitness are selected by the gambling roulette operation, which maintains the dominance relationship between individuals in the population. This dominance relationship makes the algorithm with good global convergence. Thirdly, the crossover operation is completed under the condition of population partitioning in order to achieve the purpose of dimension reduction and maintain the diversity of the population. Finally, the search direction of the grey wolf population is guided by the mutation operation of the elite individuals, which effectively prevents the algorithm from falling into the local optimal solution. All in all, through the improvement of these four different strategies, the HGGWA algorithm shows good global convergence in solving high-dimensional function optimization problems.

6. Conclusion
In order to overcome the shortcomings of GWO algorithm for solving large-scale global optimization problems which are easy to fall into local optimum, this paper proposes a Hybrid Genetic Grey Wolf Algorithm (HGGWA) by integrating three genetic operators into the algorithm. The improved algorithm initializes the population based on the Opposition-Based Learning strategy for improving the search efficiency of the algorithm. The strategy of population division based on cooperative coevolution reduces the scale of the problem, and the mutation operation of the elite individuals effectively prevents the algorithm from falling into the local optima. The performance of HGGWA has been evaluated using 10 classical benchmark functions and 7 CEC’2008 high-dimensional functions. From our experimental results, several conclusions can be drawn.

The results have shown that it is capable of grouping interacting variables with great accuracy for the majority of the benchmark functions. A comparative study among the HGGWA and other state-of-the-art algorithms was conducted and the experimental results showed that the HGGWA algorithm exhibits better global convergence whether it is solving separable functions or nonseparable functions.

(1) By testing 10 classical benchmark functions, compared with the results of WOA, SSA, and ALO, the HGGWA algorithm has been greatly improved in convergence accuracy. In addition to the Schwefel problem and Rosenbrock function, the HGGWA algorithm achieves an 80%-100% successful ratio for all benchmark functions with 100 dimensions, which proves the effectiveness of HGGWA for solving high-dimensional functions.

(2) By comparing with the optimization results on seven CEC’2008 large-scale global optimization problems, the results show that HGGWA algorithm achieves the global optimal value for the separable functions F1, F4, and F6, but the effect on other nonseparable problems is not very satisfactory. However, HGGWA algorithm still exhibits better global convergence than the other algorithms: CCPSO2, DEwSAcc, MLCC, and EPUS-PSO on most of the functions.
<table>
<thead>
<tr>
<th>Function</th>
<th>( k = 0.5 )</th>
<th>( k = 1 )</th>
<th>( k = 1.5 )</th>
<th>( k = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std</td>
<td>Mean</td>
<td>Std</td>
</tr>
<tr>
<td>( f_1 )</td>
<td>8.43E-60</td>
<td>5.36E-60</td>
<td>2.25E-54</td>
<td>2.03E-54</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>2.89E-34</td>
<td>2.31E-34</td>
<td>2.06E-33</td>
<td>1.89E-33</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>1.11E+00</td>
<td>1.02E+00</td>
<td>3.14E+01</td>
<td>2.19E+01</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>9.78E+01</td>
<td>8.36E+01</td>
<td>9.82E+01</td>
<td>7.43E+01</td>
</tr>
<tr>
<td>( f_5 )</td>
<td>2.18E-03</td>
<td>1.34E-03</td>
<td>8.87E-02</td>
<td>6.67E-02</td>
</tr>
<tr>
<td>( f_6 )</td>
<td>1.31E-03</td>
<td>1.13E-03</td>
<td>1.62E-03</td>
<td>2.43E-04</td>
</tr>
<tr>
<td>( f_7 )</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>( f_8 )</td>
<td>1.58E-14</td>
<td>1.23E-15</td>
<td>2.93E-14</td>
<td>1.37E-14</td>
</tr>
<tr>
<td>( f_9 )</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>( f_{10} )</td>
<td>2.99E-02</td>
<td>2.49E-02</td>
<td>3.38E-02</td>
<td>1.24E-03</td>
</tr>
</tbody>
</table>
(3) By analyzing the running time of several algorithms, the results show that the convergence time of HGGWA algorithm has obvious advantages compared with SSA, WOA, etc. algorithms which are proposed recently. For the functions \(f_1, f_2, f_3, f_4, f_5, f_6\), and \(f_{10}\), the running time of the HGGWA algorithm is kept between 1s and 3s under the specified convergence accuracy, which shows the excellent stability of the HGGWA.

(4) In the future, we are planning to investigate more efficient population partition mechanism to adapt to different nonseparable problems in the process of crossover operation. We are also interested in applying HGGWA to real-world problems to ascertain its true potential as a valuable optimization technique for large-scale optimization such as the setup of large-scale multilayer sensor networks [73] in the Internet of Things.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors’ Contributions

Qinghua Gu and Xuexian Li contributed equally to this work.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant Nos. 51774228 and 51404182), Natural Science Foundation of Shaanxi Province (2017JM5043), and Foundation of Shaanxi Educational Committee (17JK0425).

References


