Hindawi
Discrete Dynamics in Nature and Society
Volume 2018, Article ID 4674920, 17 pages
https://doi.org/10.1155/2018/4674920

Research Article

A Hybrid Discrete Grey Wolf Optimizer to Solve Weapon Target Assignment Problems

Jun Wang,[1] **Pengcheng Luo**,[2] **Xinwu Hu**,[1] **and Xiaonan Zhang**[1]

[1] *College of Systems Engineering, National University of Defense Technology, Changsha 410073, China*
[2] *Allsim Technology Inc., Changsha 410073, China*

Correspondence should be addressed to Jun Wang; fisher_wang@nudt.edu.cn

Received 17 June 2018; Revised 3 October 2018; Accepted 29 October 2018; Published 7 November 2018

Academic Editor: Pasquale Candito

We propose a hybrid discrete grey wolf optimizer (HDGWO) in this paper to solve the weapon target assignment (WTA) problem, a kind of nonlinear integer programming problems. To make the original grey wolf optimizer (GWO), which was only developed for problems with a continuous solution space, available in the context, we first modify it by adopting a decimal integer encoding method to represent solutions (wolves) and presenting a modular position update method to update solutions in the discrete solution space. By this means, we acquire a discrete grey wolf optimizer (DGWO) and then through combining it with a local search algorithm (LSA), we obtain the HDGWO. Moreover, we also introduce specific domain knowledge into both the encoding method and the local search algorithm to compress the feasible solution space. Finally, we examine the feasibility of the HDGWO and the scalability of the HDGWO, respectively, by adopting it to solve a benchmark case and ten large-scale WTA problems. All of the running results are compared with those of a discrete particle swarm optimization (DPSO), a genetic algorithm with greedy eugenics (GAWGE), and an adaptive immune genetic algorithm (AIGA). The detailed analysis proves the feasibility of the HDGWO in solving the benchmark case and demonstrates its scalability in solving large-scale WTA problems.

1. Introduction

As a classic military operational problem, the purpose of weapon target assignment (WTA) is to find an optimal or a satisfactory assignment solution, which determines the target attacked by each weapon, so as to maximize the total damage expectancy of hostile targets or minimize the loss expectancy of own-force assets. Since first proposed in the 1950s to support operation planning, command officers training, and weapon selection and acquisition [1], the WTA problem has attracted the attention of relevant researchers for decades. From the mathematical point of view, it is essentially an NP-complete problem [2] with nonlinear objective functions, discrete decision variables and multiple constraints. These intricate features indicate that seeking out the optimal solution for small-scale WTA problems is relatively realistic, but becomes impracticable for large-scale ones. This means in such circumstances searching for a satisfactory or suboptimal solution may be more efficient.

In the WTA study, the process of solving WTA problems can generally be divided into two phases, establishing a WTA model, and finding an optimal or a suboptimal solution for the model through an appropriate algorithm. In modeling, a variety of factors need to be considered, such as the type and quantity of equipment involved, the combat capability of each equipment, the characteristics of the battlefield, and the focus of the commander. In addition, for the convenience of modeling, certain assumptions may be made. Conventional WTA models include up to three types of entities, platforms, weapons, and targets. However, considering that weapons are increasingly dependent on the information provided by sensors in modern warfare, Bogdanowicz et al. [3, 4] believed that sensors should be considered in WTA models and established the sensor-WTA (S-WTA) model for the first time. They simplified the S-WTA problem through the sensor/target/weapon decomposition to the following scenario, each sensor-weapon pair can be assigned to at most one target, and each target can be engaged

by at most one sensor-weapon pair. Furthermore, through the sensor/weapon/target augmentation, they translated the derived problem into a symmetric optimization problem with an input consisting of the same numbers of sensors, weapons, and targets, along with the benefit matrix, and presented the Swt-opt algorithm derived from the auction algorithm to optimally assign sensors and weapons to targets. Since then, the S-WTA problem has been successively studied by some scholars. Li et al. [5, 6] proposed an improved Swt-opt algorithm to solve the same problem by combining the consensus algorithm, so that the shortcoming of the Swt-opt algorithm, i.e., highly depending on perfect network topologies, can be overcome. Later, they put forward a decentralized cooperative auction algorithm for a similar S-WTA problem where the number of targets that can be struck varied in different operational phases [7]. Chen et al. [8] proposed a particle swarm optimization based on genetic operators to solve the S-WTA problem where each sensor can guide only one weapon once and each target can be engaged by multiple weapons once. Mu et al. [9] proposed a multiscale quantum harmonic oscillator algorithm to solve the S-WTA problem in intelligent minefields considering the probability of detection and killing. Xin et al. [10] constructed an efficient marginal-return-based heuristic to solve the S-WTA problem considering the situation where multiple sensors/weapons can be assigned to one target, but each sensor can detect only one target at the same time and each weapon can shoot only one target simultaneously. The proposed heuristic exploited the marginal return of each sensor-weapon-target triplet and dynamically updated the threat value of all targets. It relied only on simple lookup operations to choose each assignment triplet, thus resulting in very low computational complexity.

Both the conventional WTA model and the S-WTA model can be classified as follows. For ease of presentation, the two models will not be distinguished, both called the WTA model. A WTA model is called to be dynamic [10–12], if the operational process is time related. This indicates there are always several operational stages or new situations usually arise during operation. Otherwise, the model is static [3–6, 8, 9, 13–16]. If two or more objectives, such as maximizing the damage expectancy of hostile targets and minimizing the consumption of own ammunition and mission completion time, are considered, the WTA model is multiobjective [17, 18]. If there is only one objective, it is single-objective [3–11, 19]. Depending on the type of combat scenario, the WTA model may be asset-based [11] or target-based [17]. In defensive missions, the asset-based model is often established to minimize the loss expectancy of own-force assets, while in offensive missions, the target-based model is always adopted to maximize the total damage expectancy of hostile targets. In addition to modeling, another research aspect for WTA is to develop algorithms to solve the problem optimally in the least possible time. These algorithms can be generally divided into two categories: exact algorithms and heuristic algorithms. Exact algorithms are developed according to the specific mathematical properties of WTA problems and can gradually eliminate the nonlinearity of the problem through transformation, decomposition, and other processing means [16]. By this way, the model is translated to a linear one, and then classical operations research methods can apply, such as the dynamic programming method [12], the branch and bound method [20, 21], the branch and price method [22], the mixed integer linear program (MILP) algorithm [17], and the Lagrange relaxation method [14]. These methods have demonstrated the feasibility and effectiveness on solving static and dynamic WTA problems but may become difficult to apply when a large number of weapons and targets are involved [23].

With the rapid development of heuristic algorithms, more complex WTA problems are able to be well solved. Most research to date on solving WTA problems by heuristic algorithms either constructs some specific search rules based on the properties of the problem to achieve solutions rapidly or introduces some local search mechanisms into the original algorithms to improve the solution quality. These algorithms, including auction algorithms [3–6, 15], improved genetic algorithms [18, 24–26], clonal selection algorithms [27, 28], particle swarm algorithms [8, 13, 29], tabu search algorithms [30], rule-based constructive heuristic algorithms [10, 31], and other intelligent optimization algorithms, have shown evident advantages over traditional methods in terms of computation time and solution accuracy and however still suffer from some drawbacks, such as easily falling into premature convergence and local optimum [32]. Furthermore, considering the variability of the battlefield environment, decisions always need to be made immediately; that is, WTA problems need to be resolved in a very short time. Therefore, we propose in this paper a hybrid discrete grey wolf optimizer (HDGWO), which is an integration of a discrete grey wolf optimizer (DGWO) with a local search algorithm (LSA). Moreover, in order to enhance the efficiency of the algorithm, we introduce the specific domain knowledge into the encode methods and the LSA.

The rest of this paper is organized as follows. The mathematical model of a typical WTA problem for an offensive mission is formulated in Section 2. Section 3 is a brief introduction to the original grey wolf optimizer (GWO) presented in [33], and Section 4 is a detailed introduction to the proposed HDGWO. In Section 5, we first analyze the feasibility of HDGWO in solving small-scale WTA problems by comparing it with the discrete particle swarm optimization (DPSO) [13], the genetic algorithm with greedy eugenics (GAWGE) [24], and the adaptive immune genetic algorithm (AIGA) [32] and then investigate the scalability of HDGWO by solving ten different scale WTA problems. Finally, we make a conclusion and look ahead to the future research in Section 6.

2. Problem Formulation

In general, a typical WTA problem for an offensive mission can be formulated as the following nonlinear integer programming model [13, 32]:

$$\max \quad f(x) = \sum_{j=1}^{N} v_j \left[1 - \prod_{i=1}^{M} \left(1 - p_{ij} \cdot e_{ij}\right)^{x_{ij}} \right]$$

Table 1: Symbol declaration.

Symbols	Explanations
M:	The number of weapon platforms that can launch missiles (e.g. fighters);
N:	The number of targets;
F_i, T_j:	The ith weapon platform and the jth target, $i = 1, 2, \cdots M$, $j = 1, 2, \cdots N$;
q_i:	The number of air-to-ground missiles available for F_i, $i = 1, 2, \cdots M$;
v_j:	The value of T_j, $j = 1, 2, \cdots N$;
p_{ij}:	The kill probability of missiles on F_i against T_j, $i = 1, 2, \cdots M$, $j = 1, 2, \cdots N$;
x_{ij}:	The decision variable indicating the number of missiles launched by F_i against T_j, $i = 1, 2, \cdots M$, $j = 1, 2, \cdots N$;
e_{ij}:	The engagement constraint factor indicating whether F_i is able to attack T_j. When one attack is feasible, $e_{ij} = 1$, otherwise, $e_{ij} = 0$, $i = 1, 2, \cdots M$, $j = 1, 2, \cdots N$;
r_j,$G_j(x)$:	The penalty factor and the penalty function, $G_j(x) = \min\{0, g_j(x)\}$, $g_j(x) = \sum_{i=1}^{M} x_{ij} - 1$, $j = 1, 2, \cdots N$;
N_p:	The population size of the proposed HDGWO;
$\vec{S}(t)$:	A solution in the tth iteration, $\vec{S}(t) = [\vec{s}_1(t), \vec{s}_2(t), \cdots, \vec{s}_M(t)]$, where $\vec{s}_i(t)$ $(i = 1, 2, \cdots M)$ is the assignment scheme of F_i. The first three best solutions are denoted by $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$ and $\vec{S}^{\delta}(t)$ in turn.
a:	The control parameter determining which rule will be adopted to update solutions;
$T_{\max}$:	The maximum iteration number;
η:	The elitist retention proportion;
Δ_p:	The perturbation value, used to modify the value of decision variables in local search;
λ:	The selection probability, determining whether a solution is to be selected for local search;

$$+ \sum_{j=1}^{N} r_j \cdot G_j(x), \tag{1}$$

subject to

$$\sum_{j=1}^{N} x_{ij} \le q_i, \quad i = 1, 2, \cdots, M, \tag{2}$$

$$\sum_{i=1}^{M} x_{ij} \ge 1, \quad j = 1, 2, \cdots, N, \tag{3}$$

$$x_{ij} \in N,$$
$$x_{ij} \ge 0, \tag{4}$$
$$i = 1, 2, \cdots, M, \ j = 1, 2, \cdots, N,$$

$$x_{ij} \cdot \left(1 - e_{ij}\right) = 0, \tag{5}$$
$$i = 1, 2, \cdots, M, \ j = 1, 2, \cdots, N.$$

The symbols in the model are explained in Table 1. For clarity, we also list the symbols employed in the following in this table.

In the above model, the objective is to maximize the damage expectancy of all targets. Thus, we construct the above objective function (1), which consists of two parts connected by a plus sign. The part before the plus sign represents the total expectancy of the eliminated value from all targets, and the latter part indicates the penalty for the solution that violates the constraint set (3). If a solution violates the constraint set (3), then the value of the latter part will become negative, which reduces the objective value. Thus, the solution will be very likely to be discarded in the next iteration. The constraint set (2) limits the number of missiles launched by each weapon platform to no more than the available number. The constraint set (4) means that decision variables are nonnegative integers. The constraint set (5) reflects the feasibility of engagement between weapon platforms and targets. When F_i is unable to attack T_j, i.e., $e_{ij} = 0$, it should not launch any missiles to T_j, i.e., $x_{ij} = 0$.

3. Grey Wolf Optimizer (GWO)

The grey wolf optimizer (GWO) is a swarm intelligence algorithm first proposed in 2014 by Seyedali Mirjalili *et al.* and mimics the leadership hierarchy and hunting mechanism of grey wolves in nature [33]. There are four types of wolves from top to bottom, denoted by α, β, δ, and ω, respectively, in the leadership hierarchy. The former three types of wolves, α, β, and δ, guide the hunting process, which is followed by the ω wolves. The GWO algorithm employs two operators to achieve optimization: (1) encircling prey and (2) hunting, as introduced below [33].

3.1. Encircling Prey. Grey wolves will first encircle prey during the hunting process. The encircling behavior is modeled by formulas (6)-(7) as follows:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right|, \tag{6}$$

$$\vec{X}(t+1) = \vec{X}(t) - \vec{A} \cdot \vec{D}, \tag{7}$$

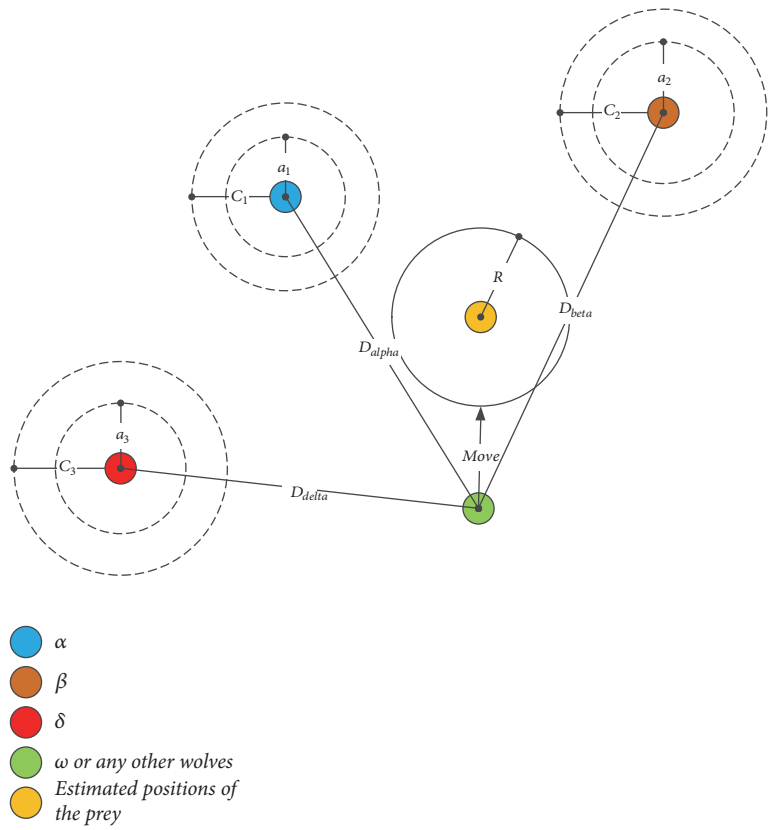

Figure 1: The position update process in GWO [33].

where t indicates the current iteration, $\vec{X}_p(t)$ and $\vec{X}(t)$ are, respectively, the position vector of the prey and the position vector of a grey wolf in the current iteration, $\vec{X}(t+1)$ is the position vector of the grey wolf in the next iteration, and $\vec{A}$ and $\vec{C}$ are coefficient vectors.

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

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}, \tag{8}$$

$$\vec{C} = 2 \cdot \vec{r}_2, \tag{9}$$

where elements of $\vec{a}$ are linearly decreased from 2 to 0 throughout the iteration process and $\vec{r}_1$ and $\vec{r}_2$ are random vectors in $[0, 1]$.

3.2. Hunting. In the GWO algorithm, the α, β, and δ wolves are supposed to have more knowledge about the potential location of prey. Therefore, the first three best solutions obtained so far (i.e., the α, β and δ wolves in the current iteration) are saved and employed to assist the other wolves in updating their positions. The process is mathematically described by formulas (10)-(12) as follows:

$$\begin{aligned} \vec{D}_\alpha &= \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \\ \vec{D}_\beta &= \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \\ \vec{D}_\delta &= \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right|, \end{aligned} \tag{10}$$

$$\begin{aligned} \vec{X}_1 &= \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha, \\ \vec{X}_2 &= \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta, \\ \vec{X}_3 &= \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta, \end{aligned} \tag{11}$$

$$\vec{X}(t+1) = \frac{\left(\vec{X}_1 + \vec{X}_2 + \vec{X}_3\right)}{3}, \tag{12}$$

where $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ are the position vectors of α, β, and δ, respectively, $\vec{C}_1$, $\vec{C}_2$, and $\vec{C}_3$ are random vectors, and $\vec{X}$ is the position vector of a grey wolf. The position update process according to α, β, and δ in a 2D search space is shown in Figure 1. The α, β, and δ wolves first estimate the position of the prey, and then other wolves update their positions randomly around the prey [33].

From formulas (6)-(12), we find that vectors $\vec{A}$ and $\vec{C}$ are two parameters to control the exploration and exploitation ability of the GWO algorithm. When $|A| > 1$, the algorithm devotes half of the iterations to explore the search space; when $|A| < 1$, the rest of the iterations are dedicated to exploitation [34]. As can be seen in formula (9), elements of $\vec{C}$ are in the interval $[0, 2]$, which provide random weights for prey in order to stochastically emphasize ($C > 1$) or deemphasize ($C < 1$) the effect of prey in determining the distance in formula (6). This mechanism gives the GWO algorithm a

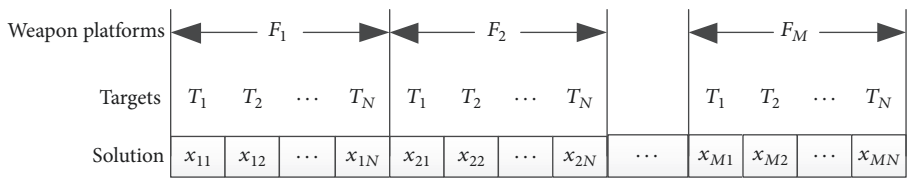

Figure 2: The decimal integer encoding method for position vectors [32].

better random search ability throughout the optimization process, which is beneficial to exploration and local optimum avoidance [33].

4. Hybrid Discrete Grey Wolf Optimizer (HDGWO) for WTA

The two operators introduced above are originally developed for solving continuous optimization problems. Thus, we cannot directly employ the GWO algorithm to address WTA problems, because the search space is discrete and decision variables are nonnegative integers. And in the field of WTA research, the application of GWO has not yet been found. For the above reasons, we first modify the original GWO to a discrete one by a decimal interencoding method and a novel position update method and then combine it with a local search algorithm (LSA). Thus, we propose a hybrid discrete grey wolf optimizer (HDGWO). The details of the HDGWO are described in the following subsections.

4.1. Decimal Integer Encoding Method. The encoding method is an important procedure before applying the proposed HDGWO to a particular problem. It constructs a bridge between solution space of the considered problem and the search space of the search process. Therefore, designing an appropriate encoding method is a very essential issue that affects the performance of the algorithm [35].

Considering that all decision variables are decimal nonnegative integers (see constraint (4) in Section 2), we adopt the decimal integer encoding method in [32] to represent the position vector of a grey wolf (in the following sections we will use the term "solution" instead of "position vector"). For the ith weapon platform F_i, we use a vector containing N elements to represent its assignment scheme, where the jth element corresponds to the number of missiles launched by F_i to T_j. So for M weapon platforms, we have M similar vectors. Then, we connect these vectors head to tail in the order of the assignment scheme of F_1 to the assignment scheme of F_M and obtain a solution, as shown in Figure 2 [32].

During the construction of an initial solution, the constraint sets (2), (4), and (5) will be handled, so the search space could be effectively compressed. We do not consider the constraint set (3) here, because on one hand, we have taken into account it when constructing the objective function (1); on the other hand the initialization method would become very complicated if the constraint set (3) should be satisfied. Although we cannot guarantee that all the solutions generated are feasible, the penalty function in the objective function (1) can help to eliminate the solution that violates the constraint set (3) throughout the iteration process. In addition, to improve the quality of initial solutions, we introduce the specific domain knowledge that available missiles should be first assigned to the target with the largest probability to be destroyed. The pseudocode of the initialization method is shown in Algorithm 1.

4.2. Fitness Function. The fitness function is used to evaluate the goodness of a solution with respect to the original objective function. For the convenience of solution comparison, we set the fitness function as the same as the objective function introduced in Section 2. The fitness value of solution S is computed as follows:

$$f(S) = \sum_{j=1}^{N} v_j \left[1 - \prod_{i=1}^{M} \left(1 - p_{ij} \cdot e_{ij}\right)^{x_{ij}(S)} \right] + \sum_{j=1}^{N} r_j \cdot G_j(S) \tag{13}$$

4.3. Searching for Prey. On the basis of the encoding method, the initialization method, and the fitness function, we can evaluate the goodness of each solution (wolf) and seek out the first three best solutions (the α, β and δ wolves). Then the next core procedure is to search for prey. In HDGWO, taking into account the constraints in Section 2 and the characteristics of the encoding method, we propose a novel position update method, namely, the modular position update method. In addition, we also adopt the elitist retention mechanism which is usually used in the genetic algorithm, and present a local search algorithm (LSA) to avoid local optimum. The details are successively introduced in the subsequent sections.

4.3.1. Modular Position Update Method. The search process of the HDGWO is also guided by the first three best solutions (wolves), denoted by $\overrightarrow{S}^{\alpha}$, $\overrightarrow{S}^{\beta}$, and $\overrightarrow{S}^{\delta}$. However, the specific position update mechanism is much different from that of the original GWO. Here we propose a modular position update method, which treats the assignment scheme of each weapon platform as a module. We first define the concept of a module as the assignment scheme of one weapon platform and then denote a solution in the tth iteration by $\overrightarrow{S}(t) = [\overrightarrow{s}_1(t), \overrightarrow{s}_2(t), \cdots, \overrightarrow{s}_M(t)]$, where $\overrightarrow{s}_i(t)$ $(i = 1, 2, \cdots, M)$ is the ith module of $\overrightarrow{S}(t)$ (i.e., the assignment scheme of

```
Input:      Combat scenario parameters, including M, N, Q = [q_i]_{1×M}, V = [v_j]_{1×N}, P = [p_ij]_{M×N}, and E = [e_ij]_{M×N},
            and the population size N_p.
Output:     The initial solutions.
Procedures: for n_p = 1 ⟶ N_p
              Set the initial value of the n_p th solution to a zero-vector [0]_{1×MN}.
              for i=1⟶M
                Sort all targets in descending order according to P(i,:). Denote the index vector by IN.
                Set q_max= q_i;
                for j=1⟶N
                  if q_max<=0 // determine if all weapons of F_i have been assigned
                    break // terminates the execution of the for loop, i.e. for j=1⟶N
                  endif
                  if E(i, IN(j))==1 // F_i can attack T_IN(j)
                      Generate a random integer, denoted by n_rand, between 1 and q_max;
                      Assign n_rand weapons to T_IN(j);
                      Update q_max= q_max-n_rand;
                  endif
                endfor
              endfor
            endfor
```

Algorithm 1: Pseudo-code of the initialization method.

F_i). Based on this, the modular position update method is formulated as follows:

$$\vec{s}_i(t+1) = \begin{cases} \vec{s}_i^{\alpha}(t) \text{ or } \vec{s}_i^{\beta}(t) \text{ or } \vec{s}_i^{\delta}(t), & \text{if } rand \le a \\ \vec{s}_i^{R}(t), & \text{otherwise} \end{cases} \quad (14)$$

where $\vec{s}_i(t+1)$ is the ith module of $\vec{S}(t+1)$ (i.e., the assignment scheme of F_i in the $(t+1)$th iteration), $\vec{s}_i^{\alpha}(t)$, $\vec{s}_i^{\beta}(t)$, and $\vec{s}_i^{\delta}(t)$, are respectively, the ith module of $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$, and $\vec{S}^{\delta}(t)$, $\vec{s}_i^{R}(t)$ is a reassignment scheme of F_i, *rand* is a random number in $(0, 1)$, and a is a control parameter. The mechanism of the modular position update method is illustrated in Figure 3, and the pseudocode of the method is described in Algorithm 2.

As can be seen from Figure 3 and Algorithm 2, the proposed HDGWO generates $\vec{s}_i(t+1)$ according to two different rules. When $rand \le a$, the roulette wheel selection method is employed to select one module among $\vec{s}_i^{\alpha}(t)$, $\vec{s}_i^{\beta}(t)$, and $\vec{s}_i^{\delta}(t)$ to be $\vec{s}_i(t+1)$ according to the fitness value of $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$, and $\vec{S}^{\delta}(t)$. This means that $\vec{s}_i^{\alpha}(t)$ acquires the largest probability to be selected, followed by $\vec{s}_i^{\beta}(t)$ and finally $\vec{s}_i^{\delta}(t)$. In this case, the assignment scheme of F_i in the tth iteration, whether $\vec{s}_i^{\alpha}(t)$, $\vec{s}_i^{\beta}(t)$, or $\vec{s}_i^{\delta}(t)$, is treated as a whole and thus will not be modified. When $rand > a$, $\vec{s}_i(t+1)$ is generated through a reassignment method, which is similar to the initialization method in Section 4.1 and is also based on the constraint sets (2), (4), and (5). Unlike the initialization method, the reassignment method directly assigns a random number of missiles to each target in the order of T_1 to T_N. Whether $\vec{s}_i(t+1)$ is generated through the roulette wheel selection method or the reassignment method, the constraint sets (2), (4), and (5) are always satisfied during the position update process.

In the proposed HDGWO, the control parameter a determines which rule will be adopted. Based on the search mechanism of the original GWO in the continuous solution space, we set a as follows [35]:

$$a = 1 - \frac{t}{T_{\max}} \quad (15)$$

where t and $T_{\max}$ are the current iteration number and the maximum iteration number, respectively. It can be seen that as t increases, a linearly decreases from 1 to 0. Therefore, the modular position update method generates $\vec{s}_i(t+1)$ mainly through the roulette wheel selection method in the first half of the iterations and thus obliges other solutions to quickly approach $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$, and $\vec{S}^{\delta}(t)$. Then in the second half, the modular position update method acquires a larger probability to generate $\vec{s}_i(t+1)$ through the reassignment method, hence giving other solutions an increasing probability to diverge from the prey. In summary, throughout the process of searching for prey, the proposed HDGWO mainly focuses on exploitation in the first half of the iterations (when $0.5 \le a \le 1$), and the second half of the iterations is mainly devoted to exploring the search space (when $0 \le a \le 0.5$).

4.3.2. Elitist Retention Mechanism. The elitist retention mechanism is employed to select a certain proportion, denoted by η, of the best solutions (wolves) in the current iteration to be a part of the next iteration directly without their positions updated. By this means, excellent solutions (wolves) are able to be preserved during the iterations

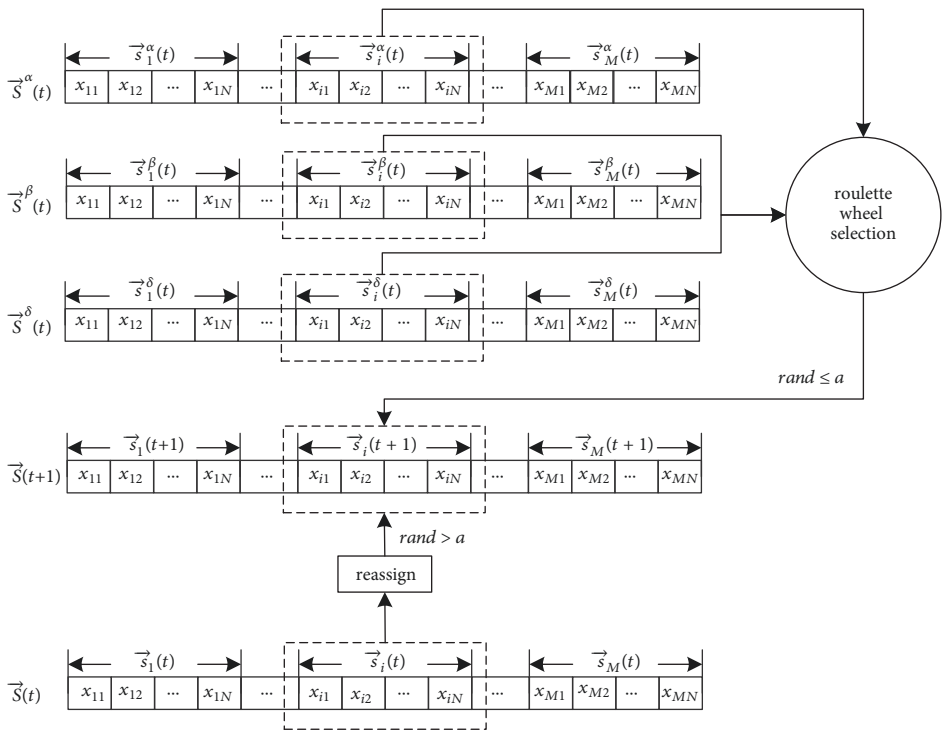

Figure 3: The mechanism of the modular position update method.

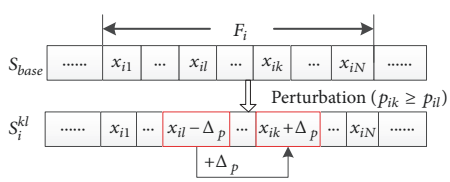

Figure 4: The mechanism of the perturbation operator.

and thus the convergence speed of the algorithm gets enhanced.

4.3.3. Local Search Algorithm (LSA). In the proposed HDGWO, we proposed a local search algorithm (LSA) based on a perturbation operator [36] to exploit the neighborhood of a solution. The mechanism of the perturbation operator is illustrated in Figure 4. Firstly randomly select a solution to be modified, denoted by S_{base}. Secondly randomly select one weapon platform F_i from the total M ones, of which the assignment scheme will be adjusted. Thirdly, randomly choose two of its targets T_k and T_l $(k, l = 1, 2, \cdots, N; k \neq l)$, satisfying $e_{ik} = e_{il} = 1$ and $x_{il}, x_{ik} > 0$. Finally, perform the perturbation operation. If $p_{ik} \geq p_{il}$, modify x_{il} to $x_{il} - \Delta_p$ and x_{ik} to $x_{ik} + \Delta_p$, and vice versa. Note that Δ_p is called the perturbation value, used to adjust the value of decision variables, and is set as 1 in this paper. After the perturbation operation, we obtain a new solution S_i^{kl} based on S_{base}.

Considering that local search is relatively time consuming, and the search process in HDGWO is guided by the α, β and δ solutions (wolves), we just apply the LSA to these three solutions with a selection probability λ that determines whether a solution is to be selected as the base solution S_{base}. The pseudocode of the LSA is described in Algorithm 3.

4.4. Programming Procedures. The pseudocode of the proposed algorithm is specified in Algorithm 4.

5. Simulation Results and Analysis

This section aims to validate HDGWO in solving WTA problems and includes two parts. In the first part, we adopt the instance in [32], an asset-based defensive WTA problem with four weapon platforms and five targets, as a benchmark case, and then compare the HDGWO with the discrete particle swarm optimization (DPSO) [13], the genetic algorithm with greedy eugenics (GAWGE) [24], and the adaptive immune genetic algorithm (AIGA) [32] by solving this benchmark case, so that the feasibility of the HDGWO to solve WTA problems could be demonstrated. To evaluate the solution quality, we also solve the case by the global solver of Lingo 11.0, a professional software package for nonlinear programming problems, and use the running result as a reference. In the second part, in order to examine the scalability of HDGWO, we first employ the above four algorithms to solve ten different scale WTA problems produced by a test case generator, and then compare the results of all algorithms from three aspects, that is, the comparison of the solution quality, the comparison of the

Input: Combat scenario parameters, including M, N, $\mathbf{Q} = [q_i]_{1\times M}$, $\mathbf{V} = [v_j]_{1\times N}$, and $\mathbf{E} = [e_{ij}]_{M\times N}$.
The first three best solutions in the tth iteration, $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$ and $\vec{S}^{\delta}(t)$.
The control parameter a.
Output: The assignment scheme of F_i in the (t+1)th iteration, $\vec{s}_i(t+1)$
Procedures: Generate a random number between (0,1), denoted by *rand*
if $rand \le a$ // the roulette wheel selection method
  Calculate the fitness value of $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$ and $\vec{S}^{\delta}(t)$, denoted by f_α, f_β and f_δ in turn;
  Calculate $w_l = f_l/(f_\alpha + f_\beta + f_\delta)$, $(l = \alpha, \beta, \delta)$;
  Set cum=0.
  Generate a random number between (0,1), denoted by r_0
  for l=α, β, δ
    cum= $cum + w_l$
    if $r_0 \le cum$
      $\vec{s}_i(t+1) = \vec{s}_i^{\,l}(t)$;
      break
    endif
  endfor
else // the reassignment method
  Set q_max= q_i;
  for j=1⟶N
    if q_max<=0 // determine if all weapons of F_i have been assigned
      break // terminates the execution of the for loop, i.e. **for** j=1⟶N
    endif
    if $\mathbf{E}(i, j))$==1 // F_i can attack T_j
      Generate a random integer, denoted by n_rand, between 1 and q_max;
      Assign n_rand weapons to T_j;
      Update q_max= q_max-n_rand;
    endif
  endfor
endif

Algorithm 2: Pseudo-code of the modular position update method.

Input: Combat scenario parameters, including M, N, $\mathbf{V} = [v_j]_{1\times N}$, $\mathbf{P} = [p_{ij}]_{M\times N}$ and $\mathbf{E} = [e_{ij}]_{M\times N}$.
The first three best solutions in the tth iteration, $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$ and $\vec{S}^{\delta}(t)$.
The selection probability λ.
Output: $\vec{S}^{\alpha}(t)$, $\vec{S}^{\beta}(t)$ and $\vec{S}^{\delta}(t)$
Procedures: **for** ls=α, β, δ
  Generate a random number between (0,1), denoted by *rand*;
  if $rand \le \lambda$ // execute the LSA
    $S_{base} = \vec{S}^{ls}(t)$;
    Generate a new solution S_i^{kl} based on the perturbation operator;
    Calculate the fitness value of S_{base} and S_i^{kl}, denoted by f_{base} and f_{new} respectively;
    if $f_{base} < f_{new}$
      Replace S_{base} with S_i^{kl}, $S_{base} = S_i^{kl}$;
    endif
    Update $\vec{S}^{ls}(t) = S_{base}$;
  endif
endfor

Algorithm 3: Pseudo-code of the local search algorithm.

Input: Combat scenario parameters, including M, N, $\mathbf{Q} = [q_i]_{1\times M}$, $\mathbf{V} = [v_j]_{1\times N}$, $\mathbf{P} = [p_{ij}]_{M\times N}$, $\mathbf{E} = [e_{ij}]_{M\times N}$.
Necessary parameters of HDGWO, including the population size N_p, the maximum iteration number $T_{\max}$, the elitist retention proportion η, the penalty factor vector $\mathbf{R} = [r_j]_{1\times N}$, the perturbation value Δ_p and the selection probability λ.

Output: The optimal or suboptimal assignment scheme.

Procedures:

Step 1. Set t =0;

Step 2. Initialize the solutions in the first iteration;

Step 3. Calculate the fitness value of all solutions in the tth iteration by formula (13), and sort solutions in descending order according to their fitness value, denoted by $\vec{S}^{(1)}(t), \vec{S}^{(2)}(t), \ldots, \vec{S}^{(N_p)}(t)$.

Step 4. Define three solution sets,
$S_{best}(t) = \{\vec{S}^{\alpha}(t), \vec{S}^{\beta}(t), \vec{S}^{\delta}(t)\} = \{\vec{S}^{(1)}(t), \vec{S}^{(2)}(t), \vec{S}^{(3)}(t)\}$,
$S_{elist}(t) = \{\vec{S}^{(i)}(t) \mid i = 1, 2, \cdots, \eta N_p\}$, and
$S_{\omega}(t) = \{\vec{S}^{(j)}(t) \mid j = \eta N_p + 1, \eta N_p + 2, \cdots, N_p\}$.

Step 5. Update $S_{\omega}(t)$ to $S_{\omega}(t+1)$ by the modular position update method;

Step 6. Perform the local search on $S_{best}(t)$, and obtain $S_{best}(t+1)$;

Step 7. Select the first N_p best solutions set $S_{current}(t+1)$ as the current solutions from $S_{best}(t+1) \cup S_{elist}(t) \cup S_{\omega}(t+1)$;

Step 8. Update t =t+1. If $t < T_{\max}$, return to **Step 3**; otherwise, continue.

Step 9. Return the solution with the best fitness value in $S_{current}(t+1)$.

Algorithm 4: Pseudo-code of HDGWO.

computation time and the comprehensive comparison based on a modified performance index (MPI). In this part, we also use the running results of Lingo 11.0 after 10000 iterations as references. All algorithms are coded in Matlab R2015a and all experimental tests are implemented on a computer with Intel Core i7-4790, 3.6 GHz, 8 GB RAM, Windows 7 operating system.

5.1. Analysis of Feasibility. Although the combat scenario in [32] is a ground-based air defensive one, the WTA model in math is essentially the same as that introduced in Section 2. The parameters to describe the combat scenario are the number of weapon platforms is M=4, the number of targets is N=5, the missile number vector is $\mathbf{Q}$=[4, 3, 2, 5], the value vector of targets is $\mathbf{V}$=[3, 1, 5, 4, 8], the engagement constraint factor matrix is $\mathbf{E}$= [1, 1, 0, 1, 0; 1, 0, 1, 1, 1; 0, 1, 1, 1, 0; 1, 1, 1, 0, 1], and the lethality probability matrix is $\mathbf{P}$= [0.2, 0.6, 0.1, 0.9, 0.7; 0.8, 0.5, 0.5, 0.7, 0.3; 0.3, 0.7, 0.8, 0.6, 0.5; 0.7, 0.5, 0.4, 0.2, 0.9]. The necessary input parameters of the HGDWO, the DPSO, the GAWGE, and the AIGA are shown in Table 2. Note that, in Table 2, we only list part of input parameters of the DPSO, the GAWGE, and the AIGA; the parameters specific to these three algorithms (i.e., the inertia weight of the DPSO, the crossover probability of the GAWGE, etc.) are consistent with those in the corresponding references and will not be listed in this paper.

We independently run each algorithm above for 100 trials and record the maximum fitness value in each trial, as illustrated in Figure 5. Then we solve the benchmark case by the global solver of Lingo 11.0. After iterating 37869 times in 13s, the global solver found the global optimal solution, as illustrated in Figure 6. The corresponding objective value was 20.736.

Table 2: The necessary input parameters of the HDGWO, the DPSO, the GAWGE, and the AIGA.

	N_p	$T_{\max}$	η	$\mathbf{r}$	Δ_p	λ
HDGWO	100	500	0.02	[1, 1, 1, 1, 1]	1	0.6
DPSO	100	500	-	[1, 1, 1, 1, 1]	-	-
GAWGE	100	500	-	[1, 1, 1, 1, 1]	-	-
AIGA	100	500	0.02	[1, 1, 1, 1, 1]	-	-

Table 3: The reference solution (assignment scheme).

	Target1	**Target2**	**Target3**	**Target4**	**Target5**
Weapon Platform1	0	**2**	0	**2**	0
Weapon Platform2	**2**	0	**1**	0	0
Weapon Platform3	0	0	**2**	0	0
Weapon Platform4	**1**	**1**	0	0	**3**

In this paper, we take the solution and the corresponding objective value obtained by Lingo 11.0 as the reference solution and the reference objective value, as shown in Table 3 and Figure 6, respectively. Then we define the successful trials of each algorithm as the trials which can obtain the reference objective value, as shown in Figure 7. Table 4 gives three statistical indicators: the average value of the maximum fitness value (AVMFV) of each algorithm for 100 trials with its standard deviation, the average value of the computation time (AVCT) of each algorithm for 100 trials with its standard deviation, and the number of the successful trials (NST).

It can be inferred from Figures 5 and 6 that the maximum fitness value is 20.736, which is consistent with that in [32]. As reflected in Figure 7 and Table 4, among the four algorithms, the proposed HDGWO wins the second largest AVMFV and NST with the least AVCT, indicating that the proposed

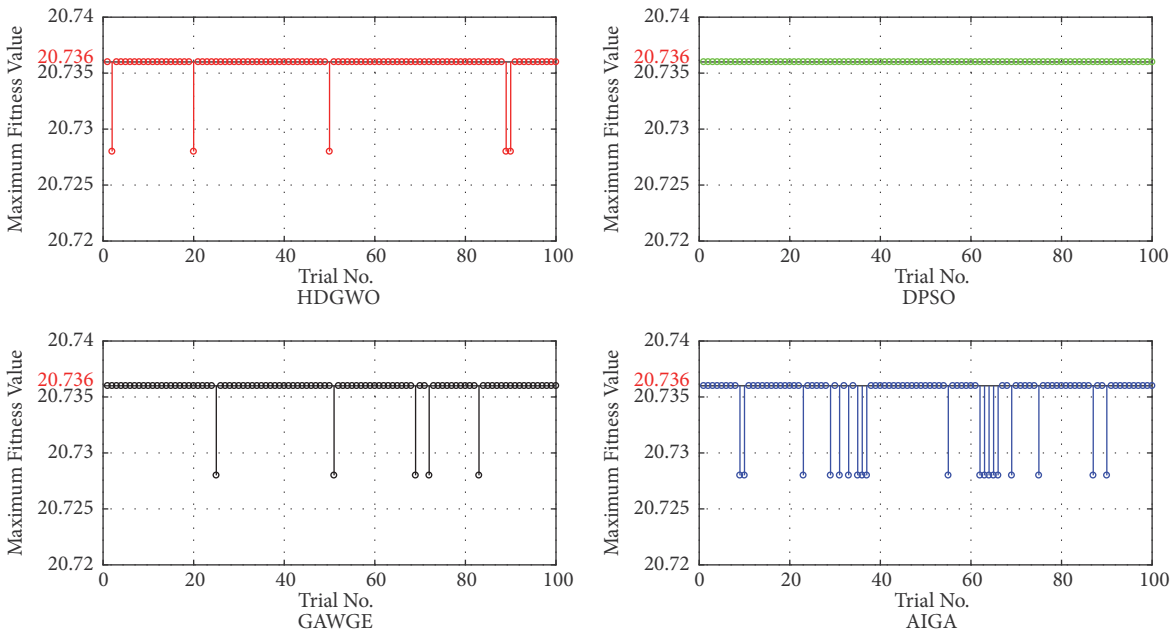

Figure 5: The maximum fitness value in each trial of the HDGWO, the DPSO, the GAWGE, and the AIGA.

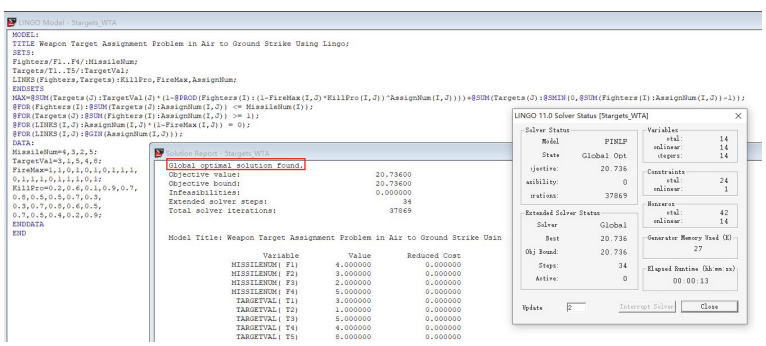

Figure 6: The running result of the global solver of Lingo 11.0.

Table 4: Three statistical indicators: the AVMFV, the AVCT, and the NST.

	HDGWO	DPSO	GAWGE	AIGA
AVMFV±std.	20.7356±0.0018	**20.7360±0.0000**	20.7356±0.0018	20.7345±0.0032
AVCT±std. (s)	**2.3760±0.0070**	19.9010±0.0290	34.8689±0.1894	3.6253±0.0170
NST	95	**100**	95	81

algorithm is able to obtain a relatively good solution in the least computation time. To be specific, in terms of the solution quality, the AVMFV of the HDGWO is, respectively, -0.002%, 0, and 0.005% higher than those of the DPSO, the GAWGE, and the AIGA. Here the negative sign "-" means the HDGWO is inferior to the DPSO. The similar superiority and inferiority can also be reflected in the NST of all algorithms. Furthermore, regarding the computation time, the AVCT of the HDGWO is about 88.1%, 93.2%, and 34.5% less than those of the DPSO, the GAWGE, and the AIGA, respectively. Therefore, it can be concluded that the HDGWO is narrowly inferior to the DPSO in terms of the solution quality, but has a great advantage over the DPSO in terms of the computation time. Therefore, we can conclude that the HDGWO proposed in this paper achieves a better trade-off between the solution quality and the computation time than the DPSO, the GAWGE, and the AIGA, and thus is feasible to solve small-scale WTA problems.

5.2. Analysis of Scalability. In this subsection, we analyze the scalability of the HDGWO by employing it to solve large-scale WTA problems and compare the running results with those of the DPSO, the GAWGE, and the AIGA. For this purpose, we first develop a test case generator to produce ten cases as introduced as follows.

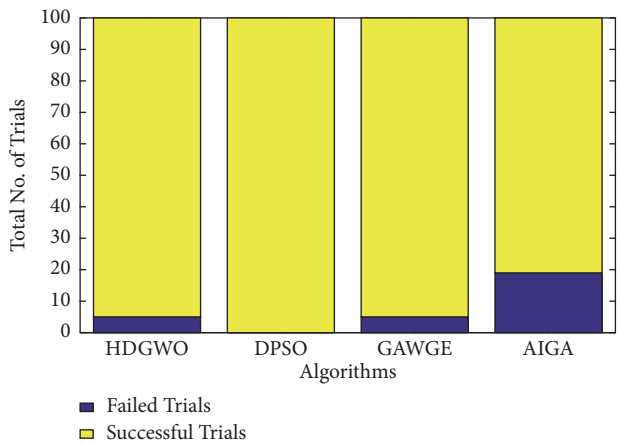

Figure 7: The number of successful trials of the HDGWO, the DPSO, the GAWGE, and the AIGA.

5.2.1. Test Case Generator

(1) Generate the number of air-to-ground missiles available for the ith weapon platform F_i as $q_i = randi(q_{\max})$, $i = 1, 2, \cdots, M$. The function $randi(x)$, where x is a positive integer, returns a pseudorandom scalar integer between 1 and x. In this paper, we set $q_{\max}$ as 8.

(2) Generate the value of the jth target T_j as $v_j = randi(v_{\max})$, $j = 1, 2, \cdots, N$. Here $v_{\max}$ is equal to 10.

(3) Generate the kill probability of missiles on F_i against T_j as $p_{ij} = p_l + (p_u - p_l) * rand$, $i = 1, \cdots, M$, $j = 1, 2, \cdots, N$. The function $rand$ returns a single uniformly distributed random number between 0 and 1, and this ensures p_{ij} between the lower bound p_l and the upper bound p_u of the lethality probability. We set $p_l = 0.1$ and $p_u = 0.9$.

(4) Generate the engagement constraint factor as $e_{ij} = [\text{sign}(rand - f_e) + 1]/2$, $i = 1, \cdots, M$, $j = 1, 2 \cdots, N$. The function $\text{sign}(x)$ returns 1 if x is a positive number, and -1 otherwise. f_e denotes the probability that e_{ij} is equal to 0, and is set as 0.2.

Ten test cases, denoted by *Case* 1 (M=20, N=20), *Case* 2 (M=40, N=40), *Case* 3 (M=60, N=60), *Case* 4 (M=80, N=80), *Case* 5 (M=100, N=100), *Case* 6 (M=120, N=120), *Case* 7 (M=140, N=140), *Case* 8 (M=160, N=160), *Case* 9 (M=180, N=180), and *Case 10* (M=200, N=200) are then generated. To reduce the computation time and improve the analysis efficiency, for all algorithms, the population size, the maximum iteration number, and the elitist retention proportion are set as 100, 500, and 0.02, respectively, while the other parameters remain unchanged.

5.2.2. Comparison of Solution Quality. We execute each algorithm for 50 trials for each case independently. Based on the running results, we then calculate the following indicators, as shown in Table 5, for the comparison of the solution quality: the average value of the maximum fitness value (AVMFV) of each algorithm for 50 trials with its standard deviation and the task completion ratio (TCR), and the improvement ratio of the solution quality (IRSQ) of one algorithm to that of another algorithm. TCR and IRSQ are defined as formula (16) and formula (17), respectively. In fact, in formula (16), the denominator should be the fitness value of the optimal solution, but it is too difficult to obtain for large-scale WTA problems. Therefore, we use the objective bounds obtained by the B-and-B (branch and bound) solver of Lingo 11.0 after 10000 iterations instead, denoted by *Lingo_ref*, which will not affect the ranking of the four algorithms on the TCR indicator. The objective bounds for each case are shown in Table 5. In addition, it may be noted that in this paper our purpose is to compare the HDGWO with the DPSO, the GAWGE, and the AIGA, so we only need to analyze the IRSQ of the HDGWO.

$$TCR = \frac{AVMFV}{Lingo\_ref} \cdot 100\%. \tag{16}$$

$$\begin{aligned} ISRQ_1 &= \frac{AVMFV_{HDGWO} - AVMFV_{DPSO}}{AVMFV_{DPSO}} \cdot 100\% \\ ISRQ_2 &= \frac{AVMFV_{HDGWO} - AVMFV_{GAWGE}}{AVMFV_{GAWGE}} \cdot 100\%. \\ ISRQ_3 &= \frac{AVMFV_{HDGWO} - AVMFV_{AIGA}}{AVMFV_{AIGA}} \cdot 100\% \end{aligned} \tag{17}$$

We can see from Table 5 that the proposed HDGWO lags behind the DPSO and the GAWGE and thus achieves the third ranking on both the AVMFV and TCR indicators in all ten cases, which indicates that it is inferior to the DPSO and the GAWGE, but superior to the AIGA in terms of the solution quality. Then we analyze the disadvantage of the HDGWO relative to the DPSO and the GAWGE though $ISRQ_1$ and $ISRQ_2$. The AVMFV of the HDGWO was only 3.07% and 5.81% smaller than those of the DPSO and the GAWGE in the worst case (i.e., in *Case 10*, $ISRQ_1$=-3.07%, $ISRQ_2$=-5.81%), respectively. And in all ten cases, the average values of $ISRQ_1$ and $ISRQ_2$ are separately -1.55% and -3.76%, indicating that the solution quality of the HDGWO is 1.55% and 3.76% worse than those of the DPSO and the GAWGE on average, respectively. Furthermore, we find that the TCR indicator of HDGWO in all cases exceeds 90% and averages 95.76%, which means that the solution obtained by the HDGWO is of satisfactory quality, even in *Case 10* which involves 200 weapon platforms and 200 targets. Therefore, based on the above analysis a conclusion can be made that the proposed HDGWO has a relatively good scalability to provide a satisfactory solution for large-scale WTA problems, even though it is a little inferior to the DPSO and the GAWGE.

5.2.3. Comparison of Computation Time. Another indicator for evaluating the performance of an algorithm is related to the computation time. In this paper, we calculate the average value of the computation time (AVCT) of each algorithm

Table 5: Three indicators of the HDGWO, the DPSO, the GAWGE and the AIGA for each case.

		Case 1	*Case 2*	*Case 3*	*Case 4*	*Case 5*	*Case 6*	*Case 7*	*Case 8*	*Case 9*	*Case 10*
Lingo_ref		***142.00***	***216.96***	***322.97***	***395.97***	***625.96***	***670.97***	***793.95***	***931.98***	***974.14***	***1060.98***
HDGWO	**AVMFV**	141.55	212.85	312.96	379.41	602.94	635.12	746.25	880.25	920.50	989.13
	(std.)	(0.57)	(3.65)	(1.99)	(5.46)	(6.40)	(6.30)	(8.46)	(7.47)	(6.21)	(6.09)
	TCR (%)	99.68	98.10	96.90	95.82	96.32	94.66	93.99	94.45	94.49	93.23
	$ISRQ_1$ (%)	-0.18	-0.55	-1.07	-1.64	-0.64	-2.23	-2.08	-2.15	-1.93	-3.07
	$ISRQ_2$(%)	-0.31	-1.80	-2.91	-3.87	-3.26	-4.81	-5.25	-4.83	-4.74	-5.81
	$ISRQ_3$ (%)	0.60	2.87	1.75	2.61	2.98	3.69	7.96	3.25	2.76	4.63
DPSO	**AVMFV**	141.81	214.03	316.34	385.73	606.82	649.57	762.12	899.56	938.58	1020.48
	(std.)	(0.03)	(0.39)	(0.25)	(0.60)	(1.10)	(0.88)	(1.48)	(1.70)	(1.89)	(1.80)
	TCR (%)	99.86	98.65	97.95	97.42	96.94	96.81	96.00	96.52	96.35	96.18
GAWGE	**AVMFV**	**141.99**	**216.75**	**322.35**	**394.69**	**623.28**	**667.18**	**787.60**	**924.89**	**966.36**	**1050.18**
	(std.)	(0.00)	(0.03)	(0.05)	(0.11)	(0.19)	(0.27)	(0.29)	(0.34)	(0.56)	(0.90)
	TCR (%)	**99.99**	**99.90**	**99.81**	**99.68**	**99.57**	**99.44**	**99.20**	**99.24**	**99.20**	**98.98**
AIGA	**AVMFV**	140.71	206.91	307.57	369.77	585.51	612.54	691.21	852.55	895.76	945.37
	(std.)	(0.77)	(3.81)	(1.80)	(6.67)	(8.06)	(7.01)	(7.11)	(8.78)	(7.24)	(10.91)
	TCR (%)	99.09	95.37	95.23	93.38	93.54	91.29	87.06	91.48	91.95	89.10

for 50 trials in all ten cases and the reduction ratio of the computation time (RRCT) of one algorithm to that of another algorithm by formula (18), as shown in Table 6. Considering our purpose in this paper, we only calculate the RRCT of the HDGWO.

As reflected in Table 6, on the AVCT indicator, the HDGWO is ranked second, just behind the AIGA in the first five cases (from *Case 1* to *Case 5*), while it achieves the first ranking in the last five cases (from *Case 6* to *Case 10*). This means that, in the aspect of the computation time, the HDGWO is inferior to the AIGA in the first five cases, but is superior to the AIGA in the last five cases and both the DPSO and the GAWGE in all cases. Since the HDGWO is inferior to both the DPSO and the GAWGE in terms of the solution quality, then we mainly investigate the advantage of the HDGWO over the DPSO and the GAWGE in the aspect of the computation time in this subsection. Note that a negative RRCT indicates HDGWO is less time consuming. By analyzing $RRCT_1$ and $RRCT_2$ we can easily find that the AVCT of the HDGWO is at least 75% smaller than those of the DPSO and the GAWGE, and on average it is 82.87% and 89.07% smaller, respectively. Thus, it can be inferred that on average the HDGWO consumes less than one-fifth of the computation time of the DPSO and one-ninth of the computation time of the GAWGE, but only reduces the solution quality by 1.55% and 3.76%, respectively. This means that the advantage of the HDGWO over the DPSO and the GAWGE in terms of the computation time is much more significant than the advantages of the DPSO and the GAWGE over the HDGWO in the aspect of the solution quality.

$$\begin{aligned} RRCT_1 &= \frac{AVCT_{HDGWO} - AVCT_{DPSO}}{AVCT_{DPSO}} \cdot 100\% \\ RRCT_2 &= \frac{AVCT_{HDGWO} - AVCT_{GAWGE}}{AVCT_{GAWGE}} \cdot 100\%. \\ RRCT_3 &= \frac{AVCT_{HDGWO} - AVCT_{AIGA}}{AVCT_{AIGA}} \cdot 100\% \end{aligned} \tag{18}$$

5.2.4. Comprehensive Comparison Based on MPI. In this subsection, we make a comprehensive comparison based on the modified performance index (MPI), which is derived from the performance index (PI) defined by Bharti [37]. The PI can be used to test the relative performance of different algorithms. Later, Mohan and Deep et al. [38–41] adopted the indicator to compare the performance of algorithms in their research. The PI of an algorithm is calculated as follows:

$$PI = \begin{cases} \dfrac{1}{num\_p} \displaystyle\sum_{j=1}^{num\_p} \left(k_1 \frac{Sr_j}{Tr_j} + k_2 \frac{Mt_j}{At_j} + k_3 \frac{Mf_j}{Af_j} \right) & \text{if } Sr_j > 0 \\ 0 & \text{if } Sr_j = 0, \end{cases} \tag{19}$$

where num_p is the total number of problems solved by the algorithm, Sr_j is the number of successful runs of the jth problem, Tr_j is the total number of runs of the jth problem, Mt_j is the minimum of the average computation time used by all algorithms in obtaining the solution for jth problem, At_j is the average computation time used by an algorithm in obtaining the solution for the jth problem, Mf_j is the minimum of the average number of function evaluations of successful runs used by all algorithms in obtaining the solution for the jth problem, Af_j is the average number of function evaluations of successful runs used by the algorithms in obtaining the solution for the jth problem, and k_1, k_2, and k_3 ($k_1 + k_2 + k_3 = 1$ and $0 \le k_1, k_2, k_3 \le 1$) are the weights assigned to the percentage of success, the computation time, and the average number of function evaluations of successful runs, respectively.

Since it is too difficult to obtain the optimal solutions for large-scale WTA problems, the number of successful runs of the jth problem Sr_j is equal to 0, indicating that PI is equal to 0. Therefore, the original PI cannot be directly used to compare the performance of the four algorithms. Considering that the performance of an algorithm needs to be comprehensively evaluated in terms of both the solution quality and the computation time, we modify the original PI from the perspective of multiattribute decision making, where the solution quality is a benefit-type attribute and the computation time is a cost-type attribute. Then we normalize the values of the two attributes to the $[0, 1]$ interval and aggregate the normalized values in the way similar to the formula (19). The modified performance index (MPI) of the ith algorithm is calculated as follows:

$$MPI_i = \frac{1}{num\_p} \sum_{j=1}^{num\_p} \left(k_1 \frac{AVMFV_i^j - AVMFV^-}{(AVMFV^+ - AVMFV^-)} + k_2 \frac{\left(AVCT^+ - AVCT_i^j\right)}{(AVCT^+ - AVCT^-)} \right), \tag{20}$$

where $AVMFV^+ = \max_i\{AVMFV_i^j\}$, $AVMFV^- = \min_i\{AVMFV_i^j\}$, $AVCT^+ = \max_i\{AVCT_i^j\}$, $AVCT^- = \min_i\{AVCT_i^j\}$, num_p is the total number of problems solved by the ith algorithm, $AVMFV_i^j$ is the $AVMFV$ of the ith algorithm in solving the jth problem, $AVCT_i^j$ is the $AVCT$ of the ith algorithm in solving the jth problem, and k_1 and k_2 are the weights of the solution quality and the computation time, satisfying $k_1 + k_2 = 1$ and $0 \le k_1, k_2 \le 1$. Note that $AVMFV$ and $AVCT$ are the average value of the maximum fitness value and the average value of the computation time, respectively, as defined in Sections 5.2.2 and 5.2.3.

We set $k_1 = w$ and $k_2 = 1 - w$, and then evaluate the MPIs of all algorithms for w=0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1, as shown in Figure 8.

It can be seen from Figure 8 that as the weight of the solution quality w increases, both the MPIs of the HDGWO and the AIGA decrease while those of the DPSO and the GAWGE increase. These trends are consistent with our previous analysis. The larger the weight of the solution quality w, the more obvious the advantages of the DPSO and the GAWGE over the HDGWO and the AIGA in terms of the solution quality. Since no algorithms can keep the first ranking on the MPI indicator under different w, then we further calculate the average value and the standard

Table 6: The AVCT indicators of the HDGWO, the DPSO, the GAWGE and the AIGA for each case.

		Case 1	*Case 2*	*Case 3*	*Case 4*	*Case 5*	*Case 6*	*Case 7*	*Case 8*	*Case 9*	*Case 10*
HDGWO	**AVCT (s)**	13.39	27.51	44.58	65.22	88.77	**114.78**	**141.53**	**172.10**	**205.02**	**240.45**
	(std.)	(0.02)	(0.02)	(0.1)	(0.19)	(0.08)	(0.12)	(0.30)	(0.44)	(0.38)	(0.45)
	$RRCT_1$ (%)	-87.64	-86.30	-85.53	-84.53	-83.28	-82.19	-81.23	-80.35	-79.32	-78.35
	$RRCT_2$ (%)	-91.61	-91.45	-90.52	-90.15	-89.00	-88.74	-88.37	-87.46	-87.06	-86.36
	$RRCT_3$ (%)	118.49	108.17	66.90	27.25	2.39	-13.36	-24.55	-33.73	-39.31	-43.99
DPSO	**AVCT (s)**	108.38	200.89	308.11	421.469	531.07	644.41	754.18	875.62	991.26	1110.70
	(std.)	(0.05)	(0.05)	(0.07)	(0.08)	(0.13)	(0.15)	(0.26)	(0.39)	(0.40)	(0.41)
GAWGE	**AVCT (s)**	159.60	321.89	470.36	662.13	807.18	1019.13	1216.85	1372.22	1583.85	1762.51
	(std.)	(0.07)	(0.24)	(0.32)	(0.39)	(0.41)	(0.53)	(0.59)	(0.64)	(0.70)	(0.82)
AIGA	**AVCT (s)**	**6.13**	**13.22**	**26.71**	**51.25**	**86.70**	132.48	187.58	259.70	337.82	429.28
	(std.)	(0.02)	(0.04)	(0.06)	(0.10)	(0.25)	(0.55)	(0.30)	(0.33)	(0.37)	(0.65)

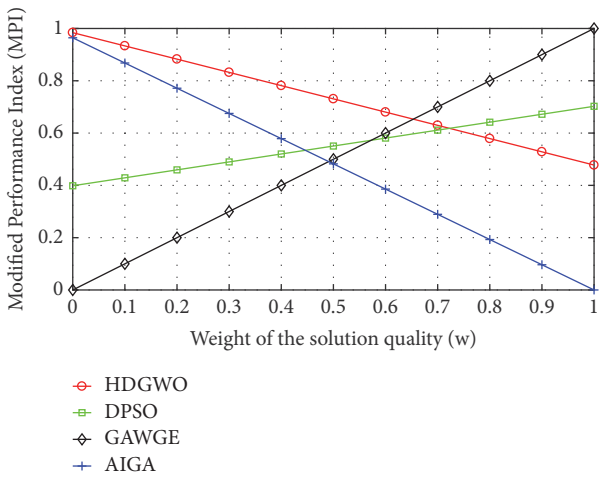

Figure 8: The MPIs of the HDGWO, the DPSO, the GAWGE, and the AIGA.

Table 7: The avg. and the std. of the MPI for the HDGWO, the DPSO, the GAWGE, and the AIGA.

	HDGWO	DPSO	GAWGE	AIGA
avg.	**0.73**	0.55	0.50	0.48
std.	0.17	0.10	0.33	0.32

deviation of the MPI for each algorithm. The average value of the MPI reflects the overall performance of the algorithm under different weights, and the standard deviation of the MPI measures the stability of the overall performance of the algorithm as the weight changes. The average value and the standard deviation of the MPI for each algorithm are shown in Table 7.

As reflected in Table 7, the average value of the MPI of the HDGWO is the largest and the corresponding standard deviation is the second smallest, only larger than that of the DPSO. This means that, from an overall point of view, the HDGWO has better performance than the other three algorithms and has the ability to maintain the performance at a relatively stable level, as the weight changes. Therefore, when the weight of the solution quality is unknown, the HDGWO should be recommended first among the four algorithms. But when the weight of the solution quality has been specified, the algorithm is recommended according to the following rules: the HDGWO is preferred if $0 \leq w < 0.6533$; otherwise, the GAWGE is preferred.

6. Conclusion and Future Research

In this paper, we present the hybrid discrete grey wolf optimizer (HDGWO) to effectively solve WTA problems on the basis of the original grey wolf optimizer (GWO), which was developed only for optimizing the problems with a continuous solution space. To make GWO available, we adopt the decimal integer encoding method to represent solutions and propose the modular position update method to update solutions in the discrete solution space. Thus, we obtain the discrete grey wolf optimizer (DGWO) and then by combining it with the local search algorithm (LSA), we acquire the HDGWO. To evaluate the feasibility of HDGWO in solving WTA problems, we first employ it to solve the benchmark case in [32] and compare the running results with those of the DPSO, the GAWGE, and the AIGA. To analyze the scalability of the HDGWO, we compare three kinds of indicators that, respectively, measure the solution quality, the computation time, and the comprehensive performance of algorithms based on the running results of the HDGWO, the DPSO, the GAWGE, and the AIGA in solving ten different scale WTA problems produced by the test case generator. The simulation results prove the feasibility of the HDGWO in solving small-scale WTA problems and demonstrate the scalability of the HDGWO in solving large-scale WTA problems. In addition, the rules for selecting the best one among the four algorithms are also provided based on the comparison of the modified performance index.

We only implement the proposed HDGWO for the static WTA problems in the current work. In future research, we will concentrate on applying HDGWO to solve more complex WTA problems, such as the dynamic problems and consider more constraints in modeling, such as the necessary damage degree to targets, dynamic engagement feasibility factors, etc. Furthermore, we will also consider to employ the HDGWO to optimize similar resource assignment problems in other areas, such as industrial production and transportation, etc.

Data Availability

The.zip data used to support the findings of this study have been deposited in the figshare repository ([https://figshare.com/articles/Data_for_HDGWO_zip/6489926]).

Conflicts of Interest

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

Authors' Contributions

Jun Wang and Pengcheng Luo conceived and designed the algorithm and experiments; Jun Wang performed the experiments; Jun Wang, Xinwu Hu, and Xiaonan Zhang analyzed the data; Jun Wang wrote the paper.

Acknowledgments

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: this work was supported by the Military Scientific Research Project (15010531150XXXX).

References

[1] A. S. Manne, "A target-assignment problem," *Operations Research*, vol. 6, pp. 346–351, 1958.

[2] S. P. Lloyd and H. S. Witsenhausen, "Weapon allocation is NP-complete," in *Proceedings of the IEEE Summer Simulation Conference*, Reno, USA, 1986.

[3] Z. R. Bogdanowicz and N. P. Coleman, "Sensor-target and weapon-target pairings based on auction algorithm," in *Proceedings of the 11th WSEAS International Conference on Applied Mathematics*, pp. 92–96, Dallas, TX, USA, 2007.

[4] Z. R. Bogdanowicz, "A new efficient algorithm for optimal assignment of smart weapons to targets," *Computers & Mathematics with Applications*, vol. 58, no. 4, pp. 1965–1969, 2009.

[5] L. Zi-fen, L. Xiang-min, D. Jin-jin, C. Jin-zhu, and Z. Feng-xia, "Sensor-weapon-target assignment based on improved SWT-opt algorithm," in *Proceedings of the 2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering (CCIE 2011)*, pp. 25–28, Wuhan, China, August 2011.

[6] Z. F. Li, X. M. Li, and J. J. Dai, "Swt-Opt Algorithm for Solving Formation Air to Ground Joint Fire Distribution Problem," *Modern Defense Technology*, vol. 40, no. 4, pp. 113-114, 2012.

[7] Z. F. Li, X. M. Li, and J. Z. Chen, "Dynamic joint fire distribution method based on decentralized cooperative auction algorithm," *Fire Control & Command Control*, vol. 37, no. 11, pp. 49–52, 2012.

[8] H. Chen, Z. Liu, Y. Sun, and Y. Li, "Particle Swarm Optimization Based on Genetic Operators for Sensor-Weapon-Target Assignment," in *Proceedings of the 2012 5th International Symposium on Computational Intelligence and Design (ISCID)*, pp. 170–173, Hangzhou, China, October 2012.

[9] L. Mu, X. Qu, and P. Wang, "Application of Sensor/Weapon-Target Assignment based on Multi-Scale Quantum Harmonic Oscillator Algorithm," in *Proceedings of the 2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pp. 1147–1151, Chengdu, China, June 2017.

[10] B. Xin, Y. P. Wang, and J. Chen, "An Efficient Marginal-Return-Based Constructive Heuristic to Solve the SensorWeaponTarget Assignment Problem," *IEEE Transactions On Systems, Man, And Cybernetics: Systems*.

[11] H. Naeem and A. Masood, "An optimal dynamic threat evaluation and weapon scheduling technique," *Knowledge-Based Systems*, vol. 23, no. 4, pp. 337–342, 2010.

[12] D. K. Ahner and C. R. Parson, "Optimal multi-stage allocation of weapons to targets using adaptive dynamic programming," *Optimization Letters*, vol. 9, no. 8, pp. 1689–1701, 2015.

[13] Y. Zhou, X. Li, Y. Zhu, and W. Wang, "A discrete particle swarm optimization algorithm applied in constrained static weapon-target assignment problem," in *Proceedings of the 2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pp. 3118–3123, Guilin, China, June 2016.

[14] M. Ni, Z. Yu, F. Ma, and X. Wu, "A Lagrange Relaxation Method for Solving Weapon-Target Assignment Problem," *Mathematical Problems in Engineering*, vol. 2011, Article ID 873292, 10 pages, 2011.

[15] Z. R. Bogdanowicz, A. Tolano, K. Patel, and N. P. Coleman, "Optimization of weapon-target pairings based on kill probabilities," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1835–1844, 2013.

[16] O. Karasakal, "Air defense missile-target allocation models for a naval task group," *Computers & Operations Research*, vol. 35, no. 2, pp. 1759–1770, 2008.

[17] N. Dirik, S. N. Hall, and J. T. Moore, "Maximizing strike aircraft planning efficiency for a given class of ground targets," *Optimization Letters*, vol. 9, no. 8, pp. 1729–1748, 2015.

[18] Y. Yan, Y. Zha, L. Qin, and K. Xu, "A research on weapon-target assignment based on combat capabilities," in *Proceedings of the 2016 IEEE International Conference on Mechatronics and Automation*, pp. 2403–2407, Harbin, Heilongjiang, China, August 2016.

[19] M. A. Şahin and K. Leblebicioğlu, "Approximating the optimal mapping for weapon target assignment by fuzzy reasoning," *Information Sciences*, vol. 255, pp. 30–44, 2014.

[20] R. K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin, "Exact and heuristic algorithms for the weapon-target assignment problem," *Operations Research*, vol. 55, no. 6, pp. 1136–1146, 2007.

[21] Y.-H. Cha and Y.-D. Kim, "Fire scheduling for planned artillery attack operations under time-dependent destruction probabilities," *Omega* , vol. 38, no. 5, pp. 383–392, 2010.

[22] O. Kwon, K. Lee, D. Kang, and S. Park, "A branch-and-price algorithm for a targeting problem," *Naval Research Logistics (NRL)*, vol. 54, no. 7, pp. 732–741, 2007.

[23] H. Cai, J. Liu, Y. Chen, and H. Wang, "Survey of the research on dynamic weapon-target assignment problem," *Journal of Systems Engineering and Electronics*, vol. 17, no. 3, pp. 559–565, 2006.

[24] Z.-J. Lee, S.-F. Su, and C.-Y. Lee, "Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 1, pp. 113–121, 2003.

[25] J. Chen, B. Xin, Z. H. Peng, L. H. Dou, and J. Zhang, "Evolutionary decision-makings for the dynamic weapon-target assignment problem," *Science in China, Series F: Information Sciences*, vol. 52, no. 11, pp. 2006–2018, 2009.

[26] Y. M. Lu, W. Q. Miao, and M. Li, "The air defense missile optimum target assignment based on the improved genetic algorithm," *Journal of Theoretical and Applied Information Technology*, vol. 48, pp. 809–816, 2013.

[27] H. Liang and F. Kang, "Adaptive chaos parallel clonal selection algorithm for objective optimization in WTA application," *Optik - International Journal for Light and Electron Optics*, vol. 127, no. 6, pp. 3459–3465, 2016.

[28] Y. Wang, W. G. Zhang, and Y. Li, "An efficient clonal selection algorithm to solve dynamic weapon-target assignment game model in UAV cooperative aerial combat," in *Proceedings of the 35th Chinese Control Conference (CCC '16)*, pp. 9578–9581, IEEE, Chengdu, China, July 2016.

[29] Y. Wang, J. Li, W. Huang, and T. Wen, "Dynamic weapon target assignment based on intuitionistic fuzzy entropy of discrete particle swarm," *China Communications*, vol. 14, no. 1, pp. 169–179, 2017.

[30] B. Xin, J. Chen, J. Zhang, L. Dou, and Z. Peng, "Efficient decision makings for dynamic weapon-target assignment by virtual permutation and tabu search heuristics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 6, pp. 649–662, 2010.

[31] B. Xin, J. Chen, Z. Peng, L. Dou, and J. Zhang, "An efficient rule-based constructive heuristic to solve dynamic weapon-target assignment problem," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 41, no. 3, pp. 598–606, 2011.

[32] S. Yang, M. Yang, S. Wang, and K. Huang, "Adaptive immune genetic algorithm for weapon system portfolio optimization in military big data environment," *Cluster Computing*, vol. 19, no. 3, pp. 1359–1372, 2016.

[33] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[34] S. Zhang, Y. Zhou, Z. Li, and W. Pan, "Grey Wolf optimizer for unmanned combat aerial vehicle path planning," *Advances in Engineering Software*, vol. 99, pp. 121–136, 2016.

[35] C. Lu, S. Xiao, X. Li, and L. Gao, "An effective multi-objective discrete grey Wolf optimizer for a real-world scheduling problem in welding production," *Advances in Engineering Software*, vol. 99, pp. 161–176, 2016.

[36] Y.-C. Liang and C.-Y. Chuang, "Variable neighborhood search for multi-objective resource allocation problems," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 3, pp. 73–78, 2013.

[37] Bharti, *Controlled random search technique and their applications [Ph.D. thesis]*, Department of Mathematics, University of Roorkee, Roorkee, India, 1994.

[38] C. Mohan and H. T. Nguyen, "A controlled random search technique incorporating the simulated annealing concept for solving integer and mixed integer global optimization problems," *Computational Optimization and Applications*, vol. 14, no. 1, pp. 103–132, 1999.

[39] K. Deep and M. Thakur, "A new mutation operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 193, no. 1, pp. 211–230, 2007.

[40] K. Deep and M. Thakur, "A new crossover operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 895–911, 2007.

[41] S. Gupta and K. Deep, "A novel Random Walk Grey Wolf Optimizer," *Swarm and Evolutionary Computation*, 2018.