Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 1424590, 11 pages
https://doi.org/10.1155/2019/1424590

Research Article

A Multiobjective Particle Swarm Optimization Algorithm Based on Multipopulation Coevolution for Weapon-Target Assignment

Guangyuan Fu, Chao Wang, Daqiao Zhang, Jiufen Zhao, and Hongqiao Wang

Xi'an Institute of High-Tech, Xi'an 710025, China

Correspondence should be addressed to Chao Wang; 18710976015@163.com

Received 2 March 2019; Revised 13 May 2019; Accepted 11 June 2019; Published 19 August 2019

Academic Editor: Thomas Hanne

Weapon-target assignment (WTA) is critical to command and decision making in modern battlefields and is a typical nondeterministic polynomial complete problem. To solve WTA problems with multiple optimization objectives, a multipopulation coevolution-based multiobjective particle swarm optimization (MOPSO) algorithm is proposed to realize the rapid search for the globally optimal solution. The algorithm constructs a master-slave population coevolution model. Each slave population corresponds to an objective function and is used to search for noninferior solutions. The master population receives all the noninferior solutions from the slave populations, repairs the gaps between the noninferior solutions, and generates a relatively optimal Pareto optimal solution set. In addition, to accelerate the slave populations searching for noninferior solutions and master population repairing the gaps between noninferior solutions, the particle velocity update method is improved. The simulation results show that the proposed algorithm has higher computational efficiency and achieves better solutions than existing algorithms capable of providing a good solution. The method is suitable for rapidly solving multiobjective WTA (MOWTA) problems.

1. Introduction

Weapon-target assignment (WTA) [1, 2] is critical to operational command and directly affects the progression and outcome of operations. WTA is an important military problem studied by numerous military powers. The core concept of WTA is to rapidly and accurately assign weapons to targets and to obtain better solutions to satisfy operational goals (i.e., optimization objectives). These optimization objectives mainly include maximum combat effectiveness, minimum weapon consumption, minimum potential threat of target, minimum target surplus, and minimum incidental damage. According to the number of optimization objectives in the problem, WTA problems can be divided into single-objective WTA (SOWTA) problems, with only one optimization objective, and MOWTA problems, with at least two optimization objectives. Currently, research on the SOWTA problem is more in-depth, and respective optimization algorithms [3–7] can quickly obtain a better allocation scheme. For the SOWTA problem with the background of air defense intercept, Liu et al. [4] proposed a firepower unit correlation matrix and designed a hybrid optimized algorithm based on the PSO and tabu search algorithm. The simulation results show that the method is more efficient than the existing methods and can output optimization results at any time. For the SOWTA problem with the background of against-ground targets, Wang et al. [5] improved the particle initialization and inertia weight selection methods of the PSO algorithm and effectively improved the optimization efficiency and allocation results of the large-scale SOWTA problem. For the dynamic SOWTA problem, Cho et al. [6] constructed a static SOWTA model with several constraints and designed an improved greedy algorithm with phased optimization, which effectively improved the optimization speed of the problem, but it was still solving a static SOWTA problem; Mei et al. [7] constructed a dynamic SOWTA model based on the killing region of weapon platform and proposed a combinatorial algorithm derived from heuristic algorithm and receding horizon control. The experimental results show that the proposed model can describe the combat scene accurately and the algorithm can improve the speed of solving the dynamic SOWTA problem. But there is still a gap from the actual application. Compared with the SOWTA problem, the MOWTA problem considers multiple optimization

objectives simultaneously, and the allocation scheme satisfies the multiple operational goals of the decision makers, therein being more in line with the combat and decision makers' needs. Research on the MOWTA problem is lacking, and the optimization speed is slow. This study researches the MOWTA problem and builds a fast optimization algorithm for such problems.

The multiobjective evolutionary algorithm based on Pareto theory [8–12] is widely used for optimization in the MOWTA problem. For the MOWTA problem with the background of against-ground targets, Li et al. [10] proposed a modified Pareto AC optimization (MPACO) algorithm based on newly designed operators, including a dynamic heuristic information calculation approach, improved movement probability rule, dynamic evaporation rate strategy, and global updating rule for pheromones. The improved MPACO algorithm has high computational efficiency and accuracy; however, the Pareto optimal solution set generated by the improved MPACO algorithm has a poor distribution and cannot satisfactorily cover the relatively optimal solutions of the WTA problem. For the MOWTA problem with the background of air defense intercept, Xia et al. [11] proposed an improved MOPSO algorithm called the MOPSO-II algorithm. By introducing a random decomposition strategy and a cooperative framework for the variables, that algorithm decomposes a large-scale-variable problem into several variable groups of moderate dimension, called subpopulations. The entire problem is then solved by cooperation between the subpopulations. Compared with the improved MPACO algorithm, the MOPSO-II algorithm has a higher optimization speed; however, the solution accuracy is slightly worse. For the dynamic MOWTA problem, Chang et al. [12] divided the dynamic process into several stages and proposed an improved ABC algorithm based on ranking selection and elite guidance. The rule-based inspiration was designed for the poor initial population of the algorithm. The simulation results show that the proposed method can improve the optimization speed and the quality of the solution, but it was still solving the static MOWTA problem. Note that the application of machine learning methods to the MOWTA problem has yet to be studied. A few scholars have studied machine learning methods used to solve the SOWTA problem [13, 14]; however, the optimization performance is not satisfactory.

The PSO algorithm is easily implemented and has a high optimization speed, also obtaining satisfactory results in the optimization of the MOWTA problem. However, the PSO algorithm is prone to premature convergence and becoming trapped in locally optimal solutions, mainly caused by a rapid loss of population diversity. To maintain population diversity, researchers have introduced operations such as random mutation [15] and stochastic perturbation [16] into the PSO algorithm or adjusted its parameters (e.g., acceleration factors [17] and inertia weights [18]). Although they maintain satisfactory population diversity and improve the global search capability of the algorithm, these improvement strategies lack an effective theoretical basis for parameter adjustment and must rely on simulation experiments, which significantly weakens their adaptability. Multipopulation strategies are another important technique to improve population diversity. Zhao et al. [19] proposed a stratified multipopulation coevolution strategy, which evenly divides the population into several subpopulations, and each subpopulation evolves in a relatively independent manner. This strategy maintains the population diversity and can effectively balance the global and local search capabilities of the algorithm. Sorkhabi et al. [20] divided the population into infeasible and feasible solution populations. These two populations evolved in parallel and independently, and the particles that evolved into feasible particles in the infeasible solution population were moved into the feasible solution population. This strategy slows the decrease in population diversity and strengthens the global search capability of the algorithm. These two multipopulation cooperation strategies provide ideas for the maintenance of population diversity and have achieved good results. However, the rate of convergence is degraded, which is unfavourable to improving the operational efficiency.

To quickly optimize the MOWTA problem, considering the advantages and disadvantages of the PSO algorithm and multipopulation cooperation strategy, a multipopulation coevolution MOPSO (MPC-MOPSO) algorithm is proposed. The MPC-MOPSO algorithm constructs a master-slave population coevolution model. A slave population corresponds to an objective function for searching noninferior solutions and speeding up the search for noninferior solutions by randomly selecting several noninferior solutions from other populations to replace particles with lower fitness values in the population. The master population receives all the noninferior solutions from the slave populations, repairs the gaps between noninferior solutions, and generates a relatively optimal Pareto optimal solution set. Through the master-slave population coevolution model, the master and slave populations can simultaneously develop different search areas of the problem. This not only maintains the good diversity of the population but also achieves the efficient coordination and global search of multiple populations. Additionally, to accelerate the process by which the populations search for noninferior solutions, repair the gaps between the noninferior solutions, and further improve the global search speed of the MPC-MOPSO algorithm, a multidirectional competition factor is introduced into the particle velocity update to guide the particles to competitively search in multiple directions. Simulation results show that, compared with the improved MPACO algorithm proposed by Li et al. [10] and the MOPSO-II algorithm proposed by Xia et al. [11], the MPC-MOPSO algorithm has higher computational efficiency and produces better solutions.

In modern and future battlefields, the benefit of assigning smart weapons to targets heavily relies on the assignment of sensors; thus, the sensor-weapon-target assignment (SWTA) problem is derived from the WTA problem. To effectively solve the SWAT problem, Bogdanowicz et al. [21] constructed a model that seeks to rationally assign sensors and weapons to targets for maximum performance. Based on Bogdanowicz et al., Wang et al. [22] modelled the damage probability as the product of the probability of target recognition by a sensor and the damage probability of a weapon paired with the sensor. Xin et al. [23] improved the damage probability model

of Wang et al. and modelled the damage probability as the product of the weapon's probability of kill and the sensor's probability of detection. They then proposed a marginal-return-based constructive heuristic (MRBCH) algorithm, which achieved good optimization results. However, the MRBCH algorithm only considers one optimization objective.

The remainder of this study is organized as follows. In Section 2, the theoretical basis of the MOWTA mode, multiobjective optimization and Pareto set, and the MOPSO algorithm are introduced and analysed. In Section 3, an MPC-MOPSO algorithm for solving the MOWTA problem is proposed. The results of employing different algorithms to solve the MOWTA problem are presented and analysed in Section 4. Section 5 concludes this study.

2. Theoretical Basis

2.1. Construction of the MOWTA Model. To correctly construct the MOWTA model, the following assumptions are made:

(1) The probability of weapon damage to the target is the comprehensive damage probability, considering the weapon's penetration probability, target hit probability, target damage probability, etc.

(2) The sequence of a target strike and the maximum projection ability of a wave of strikes are not considered.

(3) When multiple weapons strike a target, the damage probability of each weapon for the target is unchanged, and each weapon does not affect each other; that is, each strike is independent of each other.

Based on the above assumptions, the MOWTA problem can be described as follows: N types of weapons are used to attack M targets. The quantities of the N types of weapons are $N_1, N_2, \ldots, N_N$; the combat value coefficients of the M targets are V_1, V_2, …, V_M, respectively; and the comprehensive damage probability of the i^{th} type of weapon against the j^{th} target is p_{ij}, i=1, 2, …, N, j=1, 2, …, M. There are two constraints on the number of weapons used. Assuming that the number of the i^{th} type of weapon against the j^{th} target is m_{ij}, the constraints can be described as follows.

N_i is the maximum number of weapons of the i^{th} type used, i.e.,

$$\sum_{j=1}^{j=M} m_{ij} \le N_i \tag{1}$$

The attacker can use a maximum of C_j weapons on the j^{th} target, i.e.,

$$\sum_{i=1}^{i=N} m_{ij} \le C_j \tag{2}$$

The MOWTA problem has multiple optimization objectives, resulting in numerous combinations of optimization objectives. The differences in the optimization objectives and quantities of optimization objectives reflect the different operational goals of the decision makers; however, these factors do not affect the model construction technique or the optimization performance of algorithms. Therefore, this study selects two commonly used and representative optimization objectives (i.e., maximum combat effectiveness and minimum weapon consumption) to construct the model. The former seeks to maximally reduce the combat value of the target, whereas the latter seeks to destroy the target with lower cost as much as possible.

(1) Combat Effectiveness. When m_{ij} weapons of the i^{th} type attack the j^{th} target, the kill probability (S_{ij}) is

$$S_{ij} = 1 - \left(1 - p_{ij}\right)^{m_{ij}} \tag{3}$$

Thus, when all weapons of the N types attack the j^{th} target, the kill probability (S_j) is

$$S_j = 1 - \prod_{i=1}^{i=N} \left(1 - S_{ij}\right) = 1 - \prod_{i=1}^{i=N} \left(1 - p_{ij}\right)^{m_{ij}} \tag{4}$$

Let V_j be the combat value coefficient of the j^{th} target. Thus, the combat effectiveness (F_{1j}) of all weapons of the N types in attacking the j^{th} target is

$$F_{1j} = V_j S_j = V_j \left[1 - \prod_{i=1}^{i=N} \left(1 - p_{ij}\right)^{m_{ij}}\right] \tag{5}$$

The combat effectiveness (F_1) of all weapons of the N types in attacking M targets is

$$F_1 = \sum_{j=1}^{j=M} F_{1j} = \sum_{j=1}^{j=M} V_j \left[1 - \prod_{i=1}^{i=N} \left(1 - p_{ij}\right)^{m_{ij}}\right] \tag{6}$$

For the targets, F_1 is the value damage; that is, the value damage (F_2) of M targets is

$$F_2 = F_1 = \sum_{j=1}^{j=M} V_j \left[1 - \prod_{i=1}^{i=N} \left(1 - p_{ij}\right)^{m_{ij}}\right] \tag{7}$$

To rapidly and clearly compare the combat effectiveness optimization results, the coefficients V_1, V_2, …, V_M are normalized, and their conversion satisfies the following condition:

$$V = \sum_{j=1}^{j=M} V_j, \quad V_j = \frac{V_j}{V} \tag{8}$$

Thus, the combat effectiveness (F_1) of the weapons is converted to

$$F_1 = \frac{1}{\sum_{j=1}^{j=M} V_j} \sum_{j=1}^{j=M} V_j \left[1 - \prod_{i=1}^{i=N} \left(1 - p_{ij}\right)^{m_{ij}}\right] \tag{9}$$

To facilitate the use of the MOPSO algorithm in the optimization of multiple objective functions, the minimum of

the objective function (f_1=1-F_1) is used to represent the maximum F_1, i.e.,

$$f_1 = 1 - \frac{1}{\sum_{j=1}^{j=M} V_j} \sum_{j=1}^{j=M} V_j \left[1 - \prod_{i=1}^{i=N} \left(1 - p_{ij}\right)^{m_{ij}} \right] \tag{10}$$

(2) Number of Weapons Used. If m_{ij} weapons of the i^{th} type are used to attack the j^{th} target, then the number (m_i) of weapons of the i^{th} type used to attack M targets is

$$m_i = \sum_{j=1}^{j=M} m_{ij} \tag{11}$$

Thus, the number (f_2) of weapons of the N types used to attack M targets is

$$f_2 = \sum_{i=1}^{i=N} m_i = \sum_{i=1}^{i=N} \sum_{j=1}^{j=M} m_{ij} \tag{12}$$

Based on the functions f_1 and f_2 as well as the constraints on the number of weapons used, the MOWTA model is constructed as follows:

$$\begin{aligned} \min \quad & f_1 = 1 - \frac{1}{\sum_{j=1}^{j=M} V_j} \sum_{j=1}^{j=M} V_j \left[1 - \prod_{i=1}^{i=N} \left(1 - p_{ij}\right)^{m_{ij}} \right] \\ \min \quad & f_2 = \sum_{i=1}^{i=N} \sum_{j=1}^{j=M} m_{ij} \\ s.t. \quad & \sum_{j=1}^{j=M} m_{ij} \le N_i, \\ & \sum_{i=1}^{i=N} m_{ij} \le C_j \end{aligned} \tag{13}$$

2.2. Multiobjective Optimization and Pareto Set. In contrast to a single-objective optimization problem, the multiobjective optimization (MOO) problem does not have a unique optimal solution; its optimal solution becomes a set that may contain an infinite number of solutions that do not dominate one another (i.e., a Pareto optimal solution set). An MOO problem can be described as follows:

$$\begin{aligned} \min \quad & \overrightarrow{f}\left(\overrightarrow{x}\right) = \left(f_1\left(\overrightarrow{x}\right), f_2\left(\overrightarrow{x}\right), \ldots, f_n\left(\overrightarrow{x}\right)\right) \\ & \overrightarrow{g}\left(\overrightarrow{x}\right) = \left(g_1\left(\overrightarrow{x}\right), g_2\left(\overrightarrow{x}\right), \ldots, g_k\left(\overrightarrow{x}\right)\right) \end{aligned} \tag{14}$$

In (14), $\overrightarrow{x}$ is a decision vector ($\overrightarrow{x} = (x_1, x_2, \ldots, x_m)$); $\overrightarrow{f}(\overrightarrow{x})$ is an objective function vector consisting of n objective functions ($f_i(\overrightarrow{x})$); and $\overrightarrow{g}(\overrightarrow{x})$ is a constraint vector consisting of k equations ($g_i(\overrightarrow{x}) = 0$) or inequalities ($g_i(\overrightarrow{x}) \le 0$). Given an MOO problem $\overrightarrow{f}(\overrightarrow{x})$ with a feasible region $\boldsymbol{F}$, its set ($\boldsymbol{F}^*$) of Pareto optimal solutions is as follows:

$$F^* = \left\{ \overrightarrow{x} \in F \mid \neg \exists \overrightarrow{x}' \in F, \ \overrightarrow{f}\left(\overrightarrow{x}'\right) \prec \overrightarrow{f}\left(\overrightarrow{x}\right) \right\} \tag{15}$$

2.3. The MOPSO Algorithm. The MOPSO algorithm is the name of the PSO algorithm used to solve MOO problems. The MOPSO and PSO algorithms update the particle velocity and location in the same way. The PSO algorithm updates the velocity and location of a particle through two "guides": the optimal solution that the particle has found (i.e., the individual guide $\overrightarrow{p}_{best}$) and the optimal solution that the entire population has found thus far (i.e., the global guide $\overrightarrow{g}_{best}$). In the basic variant of the PSO algorithm, the velocity and location updates satisfy the following conditions:

$$\begin{aligned} v_{id}(t+1) = {} & w v_{id}(t) + r_1 c_1 \left(p_{id} - x_{id}(t)\right) \\ & + r_2 c_2 \left(g_d - x_{id}(t)\right) \end{aligned} \tag{16}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{17}$$

In (16) and (17), in the d-dimensional direction of the search space, $v_{id}(t)$ is the current velocity of the particle, $x_{id}(t)$ is the current location of the particle, $v_{id}(t+1)$ is the updated velocity of the particle, $x_{id}(t+1)$ is the updated location of the particle, p_{id} is the optimal solution that the particle has found, g_{id} is the optimal solution that the entire population has found thus far, w is the inertia weight, c_1 and c_2 are acceleration coefficients, and r_1 and r_2 are random numbers within the interval [0, 1].

The MOPSO and PSO algorithms also differ in several respects: first, the update and selection methods of the individual and global guides are different. In addition, the solution obtained by the MOPSO algorithm is a Pareto optimal solution set. Moreover, the MOPSO algorithm requires a storage set for storing the noninferior solutions found by the population to be established. Numerous researchers have conducted in-depth studies on the updating and selection of individual and global guides, the establishment of a storage set, and the parameter settings in the MOPSO algorithm and have obtained some results [24–26]. On large-scale problems, the MOPSO algorithm is still prone to premature convergence and faces difficulties in efficiently searching for the global optimal solution.

3. The MPC-MOPSO Algorithm

To maintain high population diversity and perform a rapid and global search for the Pareto optimal solutions of the MOWTA problem, the MPC-MOPSO algorithm constructs a master-slave population coevolution model, which allows the algorithm to simultaneously develop various search regions for the problem and perform an efficient multipopulation cooperative global search. Additionally, to accelerate the process in which the populations search for noninferior solutions, repair the gaps between the noninferior solutions, and further improve the global search speed of the MPC-MOPSO algorithm, a multidirectional competition factor is introduced into the particle velocity update to guide the particles to competitively search in multiple directions.

3.1. Integer Coding Based on the Attacking Target. In solving the MOWTA problem, the PSO algorithm first needs to

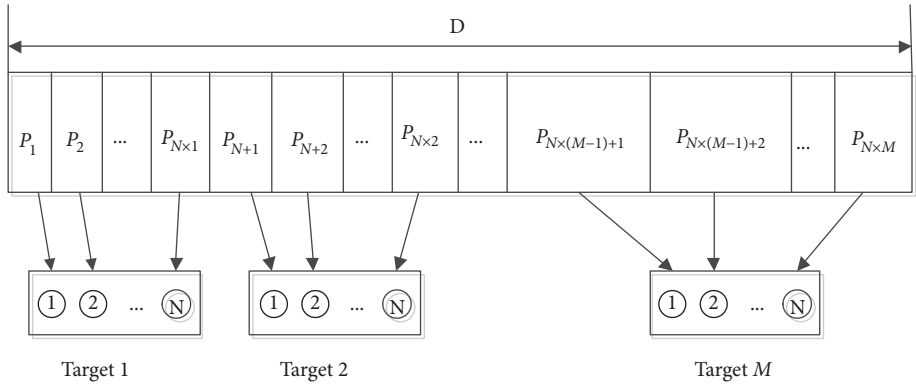

Figure 1: Diagram of the coding structure for the MOWTA problem.

encode its particles. The coding method should not only represent the solution of the problem and attempt to satisfy the constraints set in the model but also reduce the length of the code as much as possible. For this purpose, an integer coding method based on the attacking target was designed. The coding structure is shown in Figure 1. In Figure 1, the length of the code is $N\times M$; that is, the dimension is $D=N\times M$, where, P_1, P_2, …, and $P_{N\times1}$ represent the allocation of all types of weapons for target 1; P_{N+1}, P_{N+2}, …, and $P_{N\times2}$ represent the allocation of all types of weapons for target 2; and $P_{N\times(M-1)+1}$, $P_{N\times(M-2)+1}$, …, and $P_{N\times M}$ represent the allocation of all types of weapons for target M. This coding method does not distinguish the order of attack on the same target for the same type of weapon, and its weapon usage is represented by only one code, which greatly reduces the length of the code.

3.2. Particles' Multidirectional Competitive Search for Optimal Solutions. A principal factor that leads the PSO and MOPSO algorithm towards premature convergence and their low global search efficiency is described as follows. The search space contains multiple local optimal regions. The populations are guided by a rule that they tend to evolve in the direction of the optimal particle in the populations. Under this rule, the particles in the populations rapidly exchange information and tend to move in a single direction. Consequently, the populations rapidly converge to a local optimal region, and the population diversity rapidly decreases, resulting in premature convergence. If the region does not contain the globally optimal solution, the algorithm becomes trapped in local optima. The algorithm requires a substantial amount of time to jump out of local optimal solutions, resulting in low global search efficiency.

If the particles can also evolve in the directions of the optimal particles in their locally optimal regions while tending to evolve in the direction of the optimal particle, it is possible to prevent the populations from rapidly accumulating in a certain locally optimal region, thereby maintaining satisfactory population diversity, effectively preventing the algorithm from becoming trapped in locally optimal solutions and increasing the algorithm's global search speed. Hence, the particle update method is improved. A multidirectional competition factor is introduced into the particle velocity update to guide the particles to competitively search in multiple directions. The particles may still evolve in the directions of the optimal particles in their locally optimal regions even while evolving in the direction of the optimal solution. The improvement of the particle update method includes two parts: determination of the local guides and introduction of a multidirectional competition factor.

3.2.1. Determination of Local Guides. It is difficult to obtain the locally optimal regions in the search space by mathematical operations. Instead, clustering techniques can rapidly obtain more accurate locally optimal regions. Zhang et al. [27] proposed a clustering technique based on Pareto domination that can rapidly obtain accurate locally optimal regions. In this section, the Pareto-domination-based clustering technique is used to obtain locally optimal regions. "Local guides" are particles that tend to evolve in locally optimal regions and are the optimal particles in the locally optimal regions. In an MOO problem, the optimal particles are often not unique, and there may be multiple optimal particles that do not dominate one another. As a result, multiple particles may become local guides. Particle diversity can easily suffer when particles evolve in a single direction. To prevent the rapid loss of particle diversity in a locally optimal region caused by the selection of a fixed particle as the guide for all particles within a region, one particle is randomly selected from multiple optimal particles as the local guide for each particle.

3.2.2. Introduction of a Multidirectional Competition Factor into the Velocity Update. To prevent premature convergence and low global search efficiency caused by the tendency of particles to evolve in a single direction (the direction of the optimal particle in a population), a multidirectional competition factor is introduced into the particle velocity update to allow particles to evolve in the directions of the optimal particles within their locally optimal regions. This technique can realize a multidirectional competitive search by the particles, effectively preventing the algorithm from experiencing premature convergence or becoming trapped in local optima and thus improving the global search efficiency. If $\vec{l}_i$ is the local guide for particle i, then after introducing

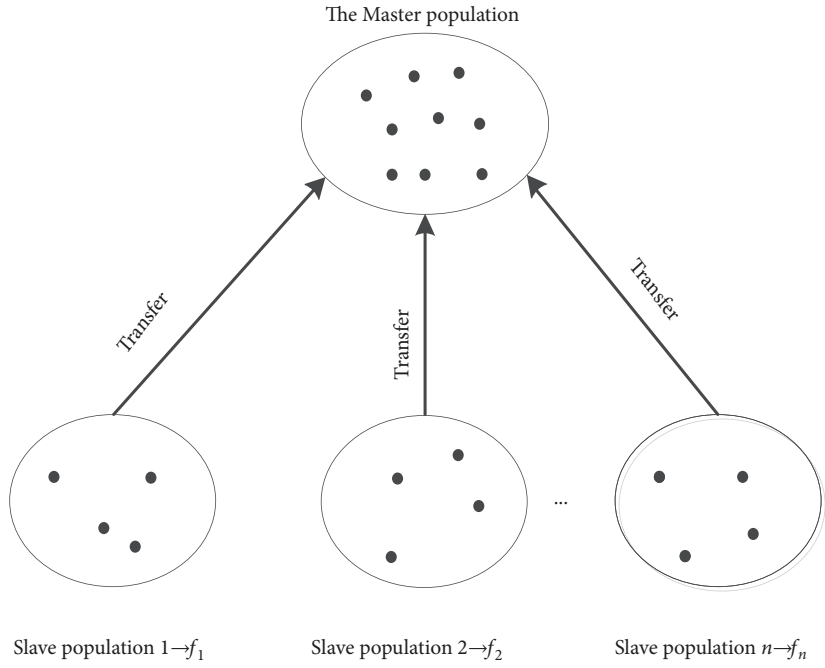

Figure 2: The master-slave population coevolution model.

a multidirectional competition factor, the velocity update shown in (16) is improved as follows:

$$v_{id}(t+1) = wv_{id}(t) + r_1c_1(p_{id} - x_{id}(t)) + r_2c_2(g_d - x_{id}(t)) + r_3c_3(l_{id} - x_{id}(t)) \quad (18)$$

In (18), c_3 is an acceleration coefficient ($c_3 = c_2/2$), r_3 is a random number in the interval $[0, 1]$, and l_{id} is the component of $\overrightarrow{l}_i$ in the d^{th} dimension of the search space.

3.3. Master-Slave Population Coevolution Model. The MPC-MOPSO algorithm constructs a master-slave population coevolution model, which includes a master population and multiple slave populations. The number of slave populations is determined by the number of objective functions. One slave population corresponds to one objective function. By implementing the PSO algorithm containing a multidirectional competition factor, the noninferior solutions of the objective function can be searched. The master population receives all noninferior solutions from the slave populations, repairs the gaps between the noninferior solutions by implementing the MOPSO algorithm containing a multidirectional competition factor, and generates a relatively optimal Pareto optimal solution set. Figure 2 shows the master-slave population coevolution model.

3.3.1. Slave Population Evolution Model. For an MOO problem with $\overrightarrow{f}(\overrightarrow{x}) = (f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_n(\overrightarrow{x}))$ as the objective functions, the slave population evolution model constructs n populations and encodes the particles in the populations using the encoding method proposed in Section 3.1. Each slave population corresponds to an objective function $f_i(\overrightarrow{x})$ (i=1, 2, …, n) and searches for an optimal solution to the corresponding objective function by using the PSO algorithm containing a multidirectional competition factor. The particles in a slave population are the decision vectors of not only the single-objective function $f_i(\overrightarrow{x})$ but also the multiobjective function $\overrightarrow{f}(\overrightarrow{x})$ and will generate the noninferior solutions needed by the master population during the iterative optimization process. Therefore, after each iterative search of a slave population, it is necessary to determine the noninferior solutions in the population and transfer them to the master population.

The slave populations operate in parallel and independently. Since the rates of evolution of the slave populations are different, some slave populations will inevitably have converged while others will have not yet converged. For slave populations that have converged, new noninferior solutions will not be generated due to the loss of population diversity. To avoid this deficiency and increase the rate of convergence of the slave populations that have not converged, a cooperative search strategy for the slave populations is proposed as follows. After each iteration of a slave population, several noninferior solutions are randomly selected from other slave populations to replace the particles with lower fitness values. The cooperative search strategy for the slave populations not only avoids the situation whereby some slave populations cannot generate new noninferior solutions due to rapid convergence but also enables the slave populations to assist one another and jointly search for noninferior solutions of single-objective functions.

3.3.2. Master Population Evolution Model. After each iteration, the master population receives all the noninferior solutions from the slave populations. Then, the population

searches for more and better noninferior solutions by the MOPSO algorithm proposed, thus repairing the gap between the noninferior solutions and generating a good Pareto optimal solution set.

Since the master population continuously receives noninferior solutions from the slave populations, its scale and particle density increase, resulting in a decrease in the optimization speed. Densely distributed particles not only are unfavourable to the maintenance of population diversity but also significantly reduce the search efficiency of the populations. Therefore, it is necessary to eliminate some densely distributed particles.

The more densely the particles are distributed, the closer the particles are to each other, and the "crowding distance" is used to measure the density of the particles. The crowding distance of a particle is the sum of its distances from its two nearest particles. If particles j and k are the particles closest to particle i, the crowding distance (D_i) of particle i is

$$D_i = \left(\sum_{d=1}^{d=D} \left(x_{id} - x_{jd} \right)^2 \right)^{-2} + \left(\sum_{d=1}^{d=D} \left(x_{id} - x_{kd} \right)^2 \right)^{-2} \quad (19)$$

Based on (19), the larger the crowding distance is, the more scattered the particles and the lower the particle density are. Additionally, the crowding distances of all particles in the master population can be obtained by using (19). If the population scale is limited to S, then the master population only retains the top S particles with the largest crowding distances.

3.4. The MPC-MOPSO Algorithm Flow. Figure 3 shows the flowchart of the MPC-MOPSO algorithm. The steps of the MPC-MOPSO algorithm are as follows:

(1) Initialize the slave populations. The number of slave populations is determined by the number of objective functions. First, encode the particles of the population following the method proposed in Section 3.1. Then, initialize the particles.

(2) Search for noninferior solutions. The slave population evolution model is used to search for noninferior solutions to the single-objective functions and move them to the master population.

(3) Generate a Pareto optimal solution set. The master population receives all the noninferior solutions from the slave populations and uses the master population evolution model to repair the gaps between the noninferior solutions. Then, a good Pareto optimal solution set is generated.

(4) Determine whether the termination condition is satisfied. If the termination condition is met, the algorithm stops searching; otherwise, the algorithm returns to Step (2).

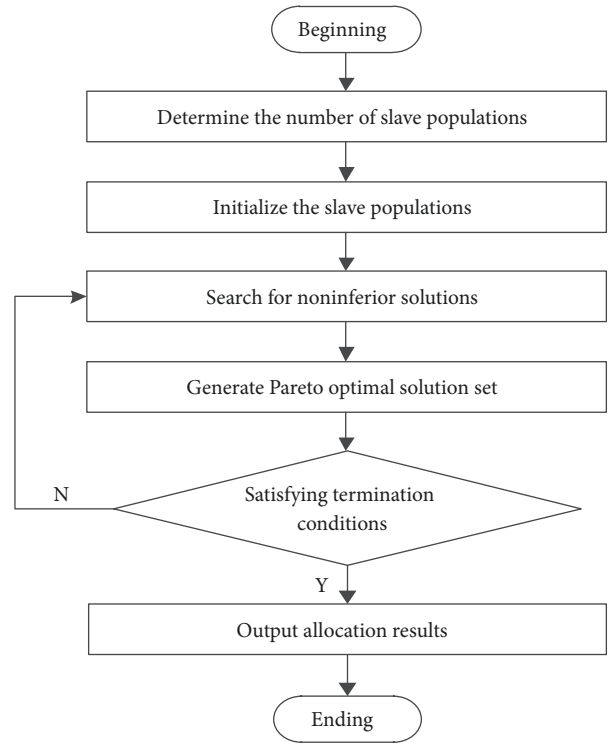

Figure 3: Flowchart of the MPC-MOPSO algorithm.

4. Simulation and Analysis of the MOWTA Problem

4.1. MOWTA Problem Cases. The case background is as follows. There are ten different types of weapons available, and twelve targets will be attacked. The number of each type of weapon, the combat value coefficient of each target, and the maximum number of weapons that can be used on each target are shown in Table 1. The comprehensive damage probability of each type of weapon for the target is shown in Table 2. Objectives with the maximum combat effectiveness and the minimum weapon consumption must be optimized.

Based on the above background, five different cases were established. The weapon types and targets in each case are shown in Table 3.

4.2. Simulation and Analysis of the MOWTA Problem. The improved MPACO algorithm proposed by Li et al. [10], the MOPSO-II algorithm proposed by Xia et al. [11], and the MPC-MOPSO algorithm proposed in this study were implemented to solve the MOWTA problem described above using the same software and hardware (64-bit Windows 7, Intel® Xeon® E3-1240 (Version 3) CPU @3.4 GHz, and 16 GB of RAM). The time (unit: s) required by each of the three algorithms to perform the optimization and the optimal assignment determined by each of the three algorithms were recorded.

4.2.1. Optimization Speeds of the Algorithms. Table 4 summarizes the time consumption statistics of the three algorithms after 20 simulations in 5 cases. $T_{\min}$ and $T_{\max}$ are the shortest and longest time of 20 simulations, respectively. T_{ave} and T_{σ} are the average time and the standard deviation of the time spent on 20 simulations, respectively.

A comparison of $T_{\min}$, $T_{\max}$, and T_{ave} for the optimizations by the three algorithms shows that the $T_{\min}$, $T_{\max}$, and T_{ave} of the MPC-MOPSO algorithm are much lower than

Table 1: Settings of weapons and targets.

Number of each type of weapon	Combat value coefficient of each target	Maximum weapon usage on each target
1, 3, 2, 2, 2, 2, 3, 1, 2, 2	8, 7, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5	2, 2, 2, 3, 1, 2, 2, 2, 2, 2, 2, 1

Table 2: Damage probability of a weapon to a target.

Weapon type	Target											
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
W_1	0.4	0.4	0.5	0.6	0.4	0.6	0.7	0.2	0.2	0.4	0.5	0.3
W_2	0.3	0.5	0.1	0.4	0.1	0.6	0.3	0.5	0.7	0.5	0.4	0.7
W_3	0.2	0.3	0.6	0.6	0.5	0.3	0.5	0.1	0.4	0.3	0.6	0.4
W_4	0.5	0.5	0.2	0.4	0.1	0.6	0.3	0.5	0.7	0.5	0.4	0.6
W_5	0.4	0.3	0.1	0.7	0.5	0.7	0.5	0.6	0.4	0.4	0.3	0.7
W_6	0.5	0.4	0.7	0.2	0.3	0.2	0.6	0.3	0.2	0.6	0.1	0.3
W_7	0.3	0.5	0.2	0.1	0.4	0.5	0.7	0.6	0.2	0.3	0.4	0.5
W_8	0.2	0.4	0.5	0.5	0.6	0.4	0.3	0.1	0.3	0.2	0.5	0.5
W_9	0.5	0.4	0.6	0.3	0.4	0.5	0.3	0.7	0.2	0.1	0.3	0.4
W_{10}	0.1	0.4	0.6	0.5	0.2	0.4	0.6	0.5	0.4	0.3	0.5	0.2

Table 3: Weapon types and targets for five cases.

Case	C_1	C_2	C_3	C_4	C_5
Weapon type	W_1-W_6	W_1-W_8	W_1-W_{10}	W_1-W_{10}	W_1-W_{10}
Target	T_1-T_8	T_1-T_8	T_1-T_8	T_1-T_{10}	T_1-T_{12}

those of the other two algorithms, suggesting that the MPC-MOPSO algorithm can significantly increase the optimization speed on MOWTA problems. Additionally, as the scale of the problem increases, T_{ave} of the MPC-MOPSO algorithm does not increase significantly compared to that of the other two algorithms, indicating that the MPC-MOPSO algorithm is suitable for solving large-scale MOWTA problems. A comparison of the T_σ of the three algorithms shows that the T_σ of the MPC-MOPSO algorithm is much lower than that of the other two algorithms, indicating that the MPC-MOPSO algorithm has good optimization stability.

4.2.2. Results Obtained Using the Algorithms. Figures 4–8 visualize the results of the Pareto optimal solution set (Pareto front, i.e., optimal numerical relationships between the maximum combat effectiveness and the minimum weapons consumption) obtained by the three algorithms in Cases 1 to 5, respectively.

Figures 4–8 show that, compared with the improved MPACO algorithm and the MOPSO-II algorithm, the Pareto front optimized by the MPC-MOPSO algorithm is closer to the true Pareto front. The comparison demonstrates that the MPC-MOPSO algorithm is accurate. Figures 4–8 also show that, with increasing problem scale, the MPC-MOPSO algorithm can still obtain good results, which shows that the MPC-MOPSO algorithm can be used to solve large-scale MOWTA problems.

For the targets, if additional weapons are put into the strike sequence, the targets will be assigned more suitable weapons. For the weapons, if additional targets are added to

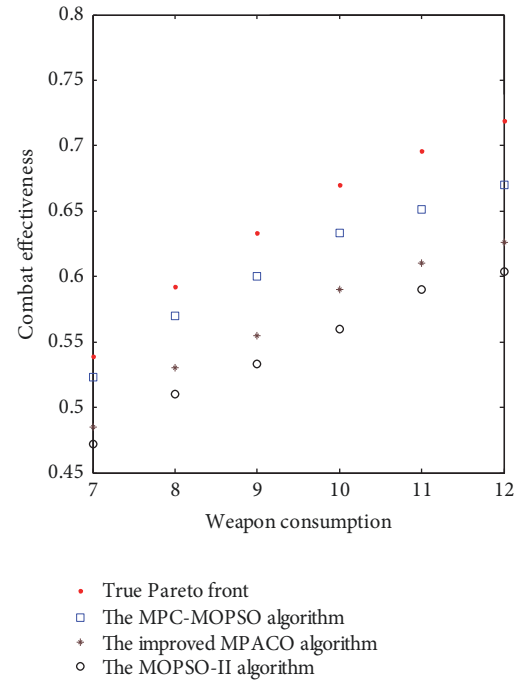

Figure 4: Pareto front in Case 1.

the strike mission, the weapons will choose more suitable targets to attack. Both of the above cases will increase the value damage of targets. To further evaluate the effectiveness, whether the MPC-MOPSO algorithm can assign more suitable weapons to the targets for the above two cases is determined. Tables 5 and 6 record the value damage of the targets under the allocation scheme obtained by the MPC-MOPSO algorithm in Case 1 to Case 3 and Case 3 to Case 5, respectively. The "N.A." in Table 5 indicates that there is no calculated value due to an insufficient number of weapons. The "N.A." in Table 6 indicates that there is no calculated

Table 4: Simulation time statistics.

Case	Number of variables	The improved MPACO algorithm				The MOPSO-II algorithm				The MPC-MOPSO algorithm			
		T_{min}	T_{max}	T_{ave}	T_σ	T_{min}	T_{max}	T_{ave}	T_σ	T_{min}	T_{max}	T_{ave}	T_σ
C_1	48	0.6	7.5	1.3	0.7	0.5	7.0	1.2	0.7	0.3	0.6	0.4	0.1
C_2	64	1.3	14.1	3.5	1.2	1.1	15.3	3.0	1.3	0.6	1.5	0.9	0.2
C_3	80	2.6	20.5	5.8	2.0	2.4	21.3	4.5	1.9	1.1	2.6	1.7	0.4
C_4	100	6.4	53.8	11.9	4.7	5.6	48.9	10.3	4.3	1.9	5.8	2.5	1.0
C_5	120	18.7	193.6	32.5	13.4	14.8	167.5	28.6	12.5	4.5	16.2	5.9	2.3

Table 5: The value damage of targets of Case 1 to Case 3.

Case	Weapon consumption					
	11	12	13	14	15	16
C_1	31.25	32.16	N.A.	N.A.	N.A.	N.A.
C_2	32.11	33.36	34.37	35.23	35.95	36.62
C_3	32.54	33.69	34.66	35.38	36.14	36.77

Table 6: The value damage of targets of Case 3 to Case 5.

Case	Weapon consumption					
	15	16	17	18	19	20
C_3	36.14	36.77	N.A.	N.A.	N.A.	N.A.
C_4	40.72	41.53	42.40	43.09	43.56	43.91
C_5	45.08	46.04	46.92	47.67	48.35	48.96

value due to the weapon usage exceeding the maximum weapon usage to targets.

Table 5 shows that, under the same weapon consumption, the value damage of targets in Case 3 is higher than that in Case 1 and Case 2, and the value damage of targets in Case 2 is higher than that in Case 1. This shows that if additional weapons are available, the MPC-MOPSO algorithm can assign weapons with a higher damage probability to the target and obtain a better allocation scheme. Table 6 shows that, under the same weapon consumption, the value damage of targets in Case 5 is higher than that in Case 3 and Case 4, and the value damage of targets in Case 4 is higher than that in Case 3. This shows that if additional targets are added to the strike mission, the MPC-MOPSO algorithm can assign targets with greater operational value to the weapons.

The comparative analysis in Section 4.2 demonstrates that the MPC-MOPSO algorithm can significantly improve the optimization speed on the MOWTA problem and generate relatively optimal assignments. Furthermore, the MPC-MOPSO algorithm can be used to solve large-scale MOWTA problems.

5. Conclusion

To quickly search for the optimal solution of the MOWTA problem, the MPC-MOPSO algorithm is proposed. The MPC-MOPSO algorithm constructs a master-slave population coevolution model and introduces a multidirectional

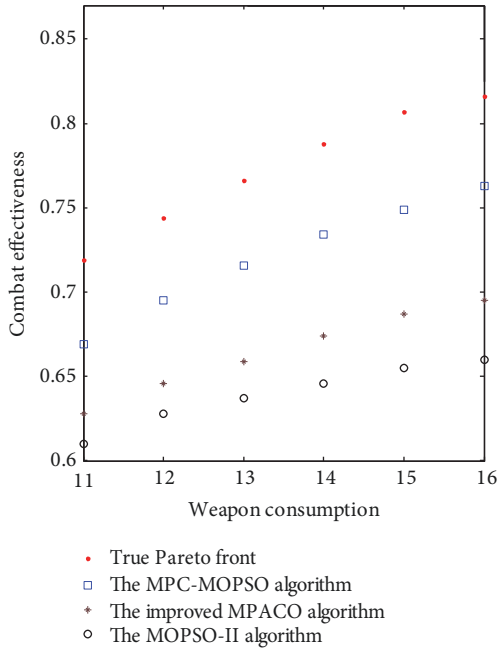

Figure 5: Pareto front in Case 2.

competition factor into the particle velocity update. The simulation results show that the MPC-MOPSO algorithm has higher computational efficiency and obtains better solutions than the improved MPACO algorithm or the MOPSO-II algorithm. The MPC-MOPSO algorithm can significantly increase the optimization speed of the MOWTA problem and generate relatively optimal assignments. Furthermore, the MPC-MOSPO algorithm can be used for rapidly solving large-scale MOWTA problems.

The MPC-MOPSO algorithm can also solve the SWTA problem. Compared with the MRBCH algorithm, which can solve the SWTA problem well, the MPC-MOPSO algorithm performs better in solving the SWTA problem with multiple optimization objectives.

Data Availability

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

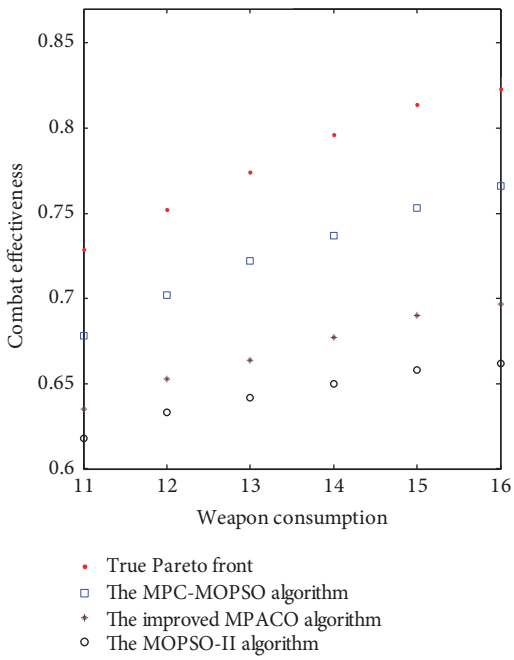

Figure 6: Pareto front in Case 3.

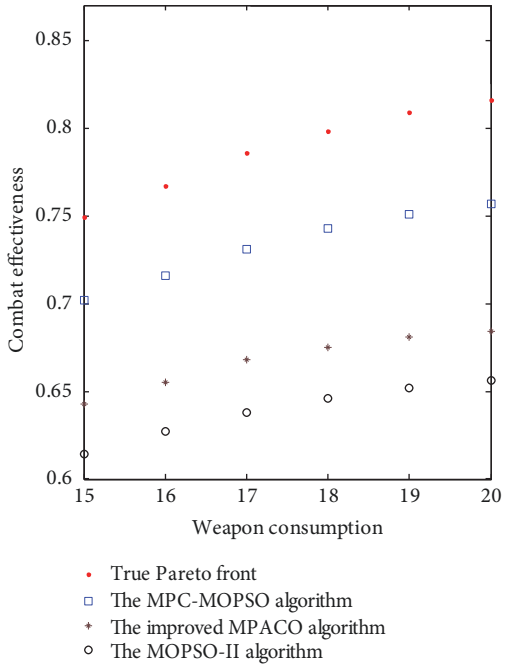

Figure 7: Pareto front in Case 4.

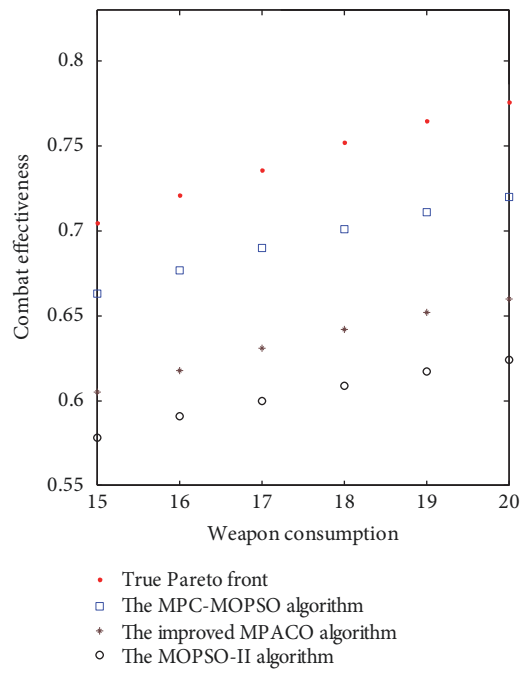

Figure 8: Pareto front in Case 5.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was jointly supported by the National Natural Science Foundation for Young Scientists of China (Grants Nos. 61403397 and 61202332) and the Natural Science Foundation of Shanxi Province, China (Grant No. 2015JM6313).

References

[1] M. F. Hocaoğlu, "Weapon target assignment optimization for land based multi-air defense systems: a goal programming approach," *Computers & Industrial Engineering*, 2019.

[2] X. Li, D. Zhou, Q. Pan et al., "Weapon-target assignment problem by multiobjective evolutionary algorithm based on decomposition," *Complexity*, vol. 2018, Article ID 8623051, 19 pages, 2018.

[3] X. Hu, P. Luo, X. Zhang et al., "Improved ant colony optimization for weapon-target assignment," *Mathematical Problems in Engineering*, vol. 2018, Article ID 6481635, 14 pages, 2018.

[4] Z. Liu, Z. Shi, L. Wu et al., "Solving cooperative anti-missile weapon-target assignment problems using hybrid algorithms based on particle swarm and tabu search," *International Conference on Computer Science and Application Engineering*, pp. 898–906, 2017.

[5] W. Shunhong, Q. Yang, R. Wang et al., "Particle Swarm Optimization based weapon-target assignment for attacking ground targets," *Electronics Optics & Control*, vol. 24, no. 3, pp. 36–40, 2017.

[6] D.-H. Cho and H.-L. Choi, "Greedy maximization for asset-based weapon-target assignment with time-dependent rewards," *Cooperative Control of Multi-Agent Systems: Theory and Applications*, pp. 115–139, 2017.

[7] Z. Mei, Z. Peng, and X. Zhang, "Optimal dynamic weapon-target assignment based on receding horizon control heuristic," in *Proceedings of the 13th IEEE International Conference on Control and Automation, ICCA 2017*, pp. 876–881, Macedonia, Balkans, July 2017.

[8] C. Feng, P. Yao, Y. Jing et al., "Weapon-target assignment based on improved quantum-inspired immune clonal multi-objective

optimization algorithm," *Computer Engineering and Science*, vol. 39, no. 12, pp. 2314–2319, 2017.

[9] Y. Zhang, Z. Qiao, and J. Jng, "Research on weapon-target allocation based on genetic algorithm," *Fuzzy Systems and Data Mining*, pp. 260–266, 2016.

[10] Y. Li, Y. Kou, Z. Li et al., "A modified pareto ant colony optimization approach to solve biobjective weapon-target assignment problem," *International Journal of Aerospace Engineering*, vol. 2017, Article ID 1746124, 14 pages, 2017.

[11] X. Xinxue, L. Yangtao, W. Fan et al., "Weapon-target assignment with an improved multi-objective particle swarm optimization algorithm," *Acta Armamentarii*, vol. 37, no. 11, pp. 2085–2093, 2016.

[12] T. Chang, D. Kong, N. Hao et al., "Solving the dynamic weapon target assignment problem by an improved artificial bee colony algorithm with heuristic factor initialization," *Applied Soft Computing*, pp. 845–863, 2018.

[13] H. Mouton, J. Roodt, and H. Le Roux, "Applying reinforcement learning to the weapon assignment problem in air defence," *Scientia Militaria: South African Journal of Military Studies*, vol. 39, no. 2, pp. 123–140, 2011.

[14] Y. Wang, J. Sun, M. Xiao et al., "Research of dynamic weapon-target assignment problem based on type-2 interval fuzzy k-nearest neighbors classifier," *Systems Engineering and Electronics*, vol. 38, no. 6, pp. 1314–1319, 2016.

[15] X. Zhang, X. Wang, Q. Kang et al., "Differential mutation and novel social learning particle swarm optimization algorithm," *Information Sciences*, vol. 480, pp. 109–129, 2019.

[16] Z. Ouyang, Y. Liu, S. Ruan et al., "An improved particle swarm optimization algorithm for reliability-redundancy allocation problem with mixed redundancy strategy and heterogeneous components," *Reliability Engineering & System Safety*, vol. 181, pp. 62–74, 2019.

[17] F. Zhou and S. Chen, "DV-Hop node localization algorithm based on improved particle swarm optimization," *Communications, Signal Processing, and Systems*, vol. 423, pp. 541–550, 2018.

[18] A. Nickabadi, R. Safabakhsh, and M. Ebadzadeh, "Inertia weight control strategies for PSO algorithms," in *Swarm Intelligence*, vol. 1 of *Principles, Current Algorithms and Methods*, pp. 169–198, 2018.

[19] N. Zhao and J. Zhao, "Self-adaptive particle swarm algorithm based on stratified multi-population," *Journal of Lanzhou University of Technology*, vol. 43, no. 4, pp. 108–111, 2017.

[20] S. A. Ebrahim, A. M. Deljavan, and K. A. Reza, "Duality evolution: an efficient approach to constraint handling in multi-objective particle swarm optimization," *Soft Computing*, vol. 21, no. 24, pp. 7251–7267, 2017.

[21] Z. Bogdanowicz and N. 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, 2007.

[22] J. Wang and C. Chen, "Sensor-weapon joint management based on improved genetic algorithm," in *Proceedings of the 34th Chinese Control Conference*, pp. 2738–2742, 2015.

[23] B. Xin, Y. Wang, and J. Chen, "An efficient marginal-return-based constructive heuristic to solve the sensor-weapon-target assignment problem," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2018.

[24] A. Monir, A. Al-Muta'a Ebtsam, and A. Sanabani Maher, "Integrated MOPSO algorithms for task scheduling in cloud computing," *Journal of Intelligent & Fuzzy Systems*, pp. 1–14, 2018.

[25] W. Zhao, Z. Luan, and C. Wang, "Parameter optimization design of vehicle E-HHPS system based on an improved MOPSO algorithm," *Advances in Engineering Software*, vol. 123, pp. 51–61, 2018.

[26] M. Najimi and S. H. R. Pasandideh, "Modelling and solving a bi-objective single period problem with incremental and all unit discount within stochastic constraints: NSGAII and MOPSO," *International Journal of Services and Operations Management*, vol. 30, no. 4, pp. 520–541, 2018.

[27] Y. Zhang and G. Gong, "Advanced multi-objective particle swarm optimization theory and its application," *Beijing City: Science Press*, pp. 77-78, 2016.