Multi-sensor target allocation model based on NC-PSO

Zhen Zhang¹,*, Jianfeng Wu¹ and Jiahao Xie¹
¹Xi'an, China

932785956@qq.com

Abstract. Traditional multi-sensor target allocation often has problems such as slow algorithm iteration and low solution quality. In response to this problem, this paper proposes the NC-PSO algorithm, so that the particles can jump out of the local optimum and increase the convergence speed. At the same time, on the basis of the improved algorithm, the constraint of the target threat degree is added, the interval analytic hierarchy process is used to estimate the target threat, and the allocation principle is optimized, so that the use of sensor resources is better and the allocation is more reasonable. The simulation results prove that NC-PSO can allocate targets to multiple sensors faster and better.

1. Introduction

In ground air defense operations, various detection sensors are deployed in a distributed manner in accordance with certain principles. They can be space-based satellites, space-based early warning aircraft, ground-based or sea-based radars, and so on. However, with the continuous development of high technology, the target gradually has the characteristics of high speed, stealth, high maneuverability, low altitude and beyond visual range strike. It is difficult for a single sensor to complete the tasks of early warning, detection, and guidance of weapon strikes. This requires the formation of detection alliances by relying on multi-sensors of land-based, sea-based, air-based, and space-based platforms, in order to improve the collaborative detection capabilities and achieve high-precision detection and tracking of targets.

In recent years, the continuous improvement of intelligent algorithms has continuously improved the performance of intelligent algorithms. Intelligent algorithms can better solve the problem of resource allocation. For example, according to the performance of the sensor, the priority of the task, and the distance between the sensor and the task execution location, T Ishak[1] et al. used the improved distributed bee algorithm (MDBA) to assign static heterogeneous sensors to the upcoming unknown task to minimize the task. Successfully minimize the number and time of task inspections. VR Sri[2] et al. used the evolutionary characteristics of genetic algorithms to continuously adjust system parameters, and at the same time introduced adaptive window sizes to limit the time delay time and ensure the latest solutions based on node movement patterns and equipment processing functions. Experiments show that the dynamic task allocation performance of the sensor network is improved. Z Sun[3] et al. used a multi-objective binary particle swarm algorithm to perform sensor-object allocation, and used nonlinear weights and elite, archiving strategies to overcome local optima and improve convergence speed. Experiments show that the algorithm can improve the efficiency of distribution. B Qi[4] et al. used fused ant colony algorithm and particle swarm algorithm to effectively solve the problem of task allocation in sensor networks. LI Tian-Long[5] et al. used the cross-mutation characteristics of genetic algorithm to optimize the update rules of particle swarm algorithm and...
effectively solved the problem of air-to-ground fire distribution. WZ Guo[6] et al. designed a binary matrix coded discrete particle swarm algorithm, which effectively solved the sensor scheduling problem. X Zhu[7] used the idea of nearest neighbor to initialize the particle task node in the particle swarm algorithm, which greatly reduced the energy consumption in the task distribution of the sensor network.

Although the methods in the literature can find the solution of the problem, the elements of the objective function are somewhat imperfect, which leads to the lack of persuasiveness of the model, and the convergence speed and the quality of the solution in the solution process need to be further improved.

In response to the above problems, this paper adds the link of target threat estimation by studying the modeling principles and conditions in the case of multi-sensor and multi-target, and improves the performance of particle swarm algorithm, and proposes a multi-sensor target allocation model based on NC-PSO. First, this paper builds a multi-sensor alliance detection model, and then introduces enhancement factors and collision factors to improve the traditional particle swarm algorithm to increase its optimization speed and the ability to jump out of local optimum, so as to improve the solution speed and accuracy of the multi-sensor detection alliance scheme.

2. Alliance model establishment

The multi-sensor detection alliance is a joint detection network composed of multiple types and numbers of sensors based on target detection functions, which can improve the utilization efficiency of sensor resources on the premise of ensuring the completion of detection tasks.

Assuming the number of sensors is \( m \) and the number of targets is \( n \), the sensor sequence for target \( j \) is \( S_j = (s_{1j}, s_{2j}, \ldots, s_{nj}) \). The sensor alliance \( S \) can be expressed as a matrix \( m \times n \). The elements \( s_{ij} \) can be expressed as:

\[
\begin{cases}
1 & \text{The sensor is in the target detection alliance} \\
0 & \text{The sensor is not in the target detection alliance}
\end{cases}
\]

2.1. Alliance accuracy model

The accuracy of the sensor \( A \) can be expressed as the ratio of the effective detection time of the sensor \( s_i \) to the target \( j \) and the target movement time, expressed as:

\[
A_{ij} = \frac{\text{time}_{ij}}{\text{Time}_{ij}}
\]  \hspace{1cm} (1)

In the formula, \( \text{time}_{ij} \) represents the effective detection time of sensor \( s_i \) to target \( j \). \( \text{Time}_{ij} \) is the flight time of target \( j \).

2.2. Sensor energy consumption model

When the sensor detects the target, it needs to consume energy. The energy consumption of the sensor to detect the target is expressed as:

\[
C(s_i) = \frac{1}{\exp\left(\frac{1}{\alpha_i \times r(s_i) + \beta_i}\right)}
\]  \hspace{1cm} (2)

In the formula, \( r(s_i) \) represents the farthest detection distance of the sensor \( s_i \), and \( \alpha_i \) and \( \beta_i \) are constants, respectively taking 0.01 and 0.1.

2.3. Target threat model

(1) Target type threat index \( O_j \)

Target types are divided into large aircraft (strong area attack capability), small aircraft (strong fixed-point attack capability), and helicopters. The threat index is set at 0.8, 0.5, and 0.3 respectively.
(2) Target distance threat index ($D_j$)
The distance threat index of target $j$ to our defensive area can be expressed as:

$$D_j = e^{-\mu_1 d_j}$$  \hspace{1cm} (3)

In the formula, $\mu = -4 \times 10^{-5} \text{km}^{-2}$, $d_j$ represents the distance between the target and our defensive area.

(3) Target height threat index ($H_j$)
The height threat index of target $j$ to our defensive area can be expressed as:

$$H_j = \begin{cases} 
1, & 0 \leq h_j \leq \eta \\
 e^{-\mu_2 (h_j - \eta)}, & \eta \leq h_j 
\end{cases}$$  \hspace{1cm} (4)

In the formula, $h_j$ is the height of target $j$, $\mu = 10^{-2} \text{km}^{-2}$, $\eta = 0.2 \text{km}$.

(4) Target speed threat index ($V_j$)
The speed threat index of target $j$ can be expressed as:

$$V_j = 1 - e^{\mu_3 |v_j|}$$  \hspace{1cm} (5)

In the formula, $\mu = -5 \times 10^{-3} \text{s/m}$, $v_j$ is the speed of target.

(5) Target route shortcut threat index ($L_j$)
The route shortcut threat index of target $j$ can be expressed as:

$$L_j = e^{-\mu_4 l_j}$$  \hspace{1cm} (6)

In the formula, $\mu = 5 \times 10^{-3} \text{km}$, $l_j$ is the shortcut of the target relative to the defensive route. Therefore, the comprehensive threat degree of target $j$ can be expressed as:

$$E_j = w_1 O_j + w_2 D_j + w_3 H_j + w_4 V_j + w_5 L_j$$  \hspace{1cm} (7)

In the formula, $w_1, w_2, w_3, w_4, w_5$ are the weight coefficients of each threat element index. The calculation of the weight is analyzed by the Interval Analytic Hierarchy Process (IAHP).

According to the obtained weight $\mathbf{W}$ and the threat index of each threat element of the target, the total threat degree of each target can be calculated, and the expression is:

$$T_j = \sum_{i=1}^{n} (\mathbf{W} \times P_i)$$  \hspace{1cm} (8)

In the formula, $\mathbf{W}$ is the weight of threat elements calculated according to the IAHP method, and $P_i$ is the calculated target threat index matrix.

2.4. Fitness function
When faced with targets with different threat levels, the multi-sensor network pays different attention to the target, allocates different sensor resources to form a multi-sensor detection alliance, so that the sensor network resource utilization efficiency is the highest, and the solution is the optimal solution. The expression of fitness value is:

$$\varphi(x) = \frac{\sum \prod \left[1 - \frac{1 - \varphi_i(x)}{\sum C_i(x)}\right]}{\sum C_i(x)}$$  \hspace{1cm} (9)
In the formula, $T_j$ is the threat degree of the target $j$, which is regarded as the degree of attention of the sensor network to the target $j$.

At the same time, assuming that each sensor can accurately track 4 targets at the same time, each $i$ row of the $S$ matrix adds up to 4 at most, that is:

$$N(i) = \sum_j s_{ij} \leq 4$$  \hspace{1cm} (10)

In the formula, $N(i)$ is the number of targets tracked by sensor $i$.

The purpose is to maximize $\psi (X)$, then the objective function is:

$$\max[\psi (X)]$$  \hspace{1cm} (11)

3. Improvement of particle swarm algorithm

3.1. Basic particle swarm algorithm

The particle swarm algorithm[8] regards the process of solving the optimal solution or approximate solution of the problem as a process of predation by a flock of birds. Each bird is equivalent to a particle, which is a feasible solution to the problem. Since the particle swarm algorithm requires fewer analysis parameters and variables in the solution process, and has a strong local search ability, this paper uses the particle swarm optimization algorithm to solve the problem of selecting members of the sensor alliance.

The particle swarm algorithm sets a population composed of $n$ particles, its position is $X_i = (X_{i1}, X_{i2}, ..., X_{id})^T$, velocity is $V_i = (V_{i1}, V_{i2}, ..., V_{id})^T$, the individual extreme value is $P_i = (P_{i1}, P_{i2}, ..., P_{id})^T$, and the global extreme value is $P_g = (P_{g1}, P_{g2}, ..., P_{gd})^T$. In each iteration, the particle adjusts its speed and position according to the positions of $P_i$ and $P_g$, so that the particle can approach $P_g$, and finally finds the optimal solution or approximate solution. The update formula for its speed and position is:

$$\begin{align*}
V_{id}^{k+1} &= wV_{id}^k + c_1r_1(P_{id}^k - X_{id}^k) + c_2r_2(P_{gd}^k - X_{id}^k) \\
X_{id}^{k+1} &= X_{id}^k + V_{id}^{k+1}
\end{align*}$$  \hspace{1cm} (12)

In the formula, $c_1$ and $c_2$ are learning factors, generally random numbers uniformly distributed between $(0,2)$, $r_1$ and $r_2$ are random numbers between $(0,1)$, and $w$ is the inertia weight.

3.2. Improved particle swarm algorithm

In the basic particle swarm algorithm, although the algorithm has few parameters and strong local search ability, it has the problem that the particles are easy to fall into the local optimum and the speed of convergence is slow.

In response to this problem, the negative reinforcement factors $\alpha_1$, $\alpha_2$ are introduced to enable the particles to move away from the current poor solutions of individuals and groups, indirectly narrowing the optimization range and increasing the speed of optimization. At the same time, the collision factors $c_3r_3$, $c_4r_4$ are introduced. When particles fall into the local optimum and lose their vitality, collisions can increase the diversity of the particles so that the particles still have the ability to search while maintaining the current optimum solution, which helps to avoid falling into the local optimum. The particle velocity update formula becomes:
\[ V_{i}^{t+1} = wV_{i}^{t} + c_{1}r_{1}(P_{i}^{t} - X_{i}^{t}) + c_{2}r_{2}(P_{g}^{t} - X_{i}^{t}) - a_{1}(P_{d}^{t} - X_{i}^{t}) - a_{2}(P_{a}^{t} - X_{i}^{t}) \] (13)

In the formula, \( c_{3} \) and \( c_{4} \) represent the impact strength of the collision factor, and the range of the collision strength can be determined according to the range of motion of the particle in the specific problem, \( r_{1} \) and \( r_{2} \) are random numbers between \((0,1)\); \( P_{d}^{t} \) is the worst solution in individual history, and \( P_{a}^{t} \) is the worst solution in population history; \( a_{1} \) is a random number uniformly distributed on \((0, a_{j, max})\), and each \( a_{j} \) is an adjustable parameter and a positive number; the values of \( c_{1}r_{1} \) and \( c_{2}r_{2} \) can be calculated according to the following formula:

\[
\begin{align*}
    c_{1}r_{1} &= \begin{cases} 
        1, & e_{p} > e_{g} \\
        c_{1}r_{1}, & e_{p} \leq e_{g}
    \end{cases} \\
    c_{2}r_{2} &= \begin{cases} 
        1, & e_{g} > e_{s} \\
        c_{2}r_{2}, & e_{g} \leq e_{s}
    \end{cases}
\end{align*}
\] (14)

In the formula, \( e_{p} \), \( e_{s} \), \( e_{g} \) are the deviation between the current fitness value and the individual extreme value and the group extreme value respectively; \( e_{p} \) and \( e_{g} \) are the thresholds of deviation.

When the deviation is greater than the threshold, the collision factor is assigned to change the current position of the particle, and the fitness value is calculated. If the fitness value of the new position is greater than the historical optimal value, the position is replaced, otherwise. Otherwise, continue to collide, calculate and compare fitness values, until the end of the iteration.

4. Simulation

4.1. Simulation conditions

Suppose that in an air defense operation, there are 7 targets, and the defender has 10 sensors join the detection mission. The target parameters and parameter indexes are shown in Table 1, and the detection accuracy and energy consumption of the sensor to the target are shown in Table 2.

| serial number | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---------------|---|---|---|---|---|---|---|
| type          | large aircraft | large aircraft | small aircraft | helicopter | small aircraft | small aircraft | small aircraft |
| speed(m/s)    | 400 | 270 | 680 | 70 | 500 | 290 | 410 |
| Height(km)    | 10 | 4.5 | 8 | 1 | 0.2 | 1 | 5 |
| route shortcut(km) | 9 | 7 | 7 | 10 | 5 | 5 | 12 |
| Distance(km)  | 150 | 80 | 180 | 50 | 160 | 70 | 210 |
| type index    | 0.8 | 0.8 | 0.5 | 0.3 | 0.5 | 0.5 | 0.5 |
| speed index   | 400 | 270 | 680 | 70 | 500 | 290 | 410 |
| height index  | 10 | 4.5 | 8 | 1 | 0.2 | 1 | 5 |
| route shortcut index | 9 | 7 | 7 | 10 | 5 | 5 | 12 |
| distance index | 150 | 80 | 180 | 50 | 160 | 70 | 210 |

Table 2. The sensor's target detection accuracy and energy consumption table.

| Sensor | Target 1 | Target 2 | Target 3 | Target 4 | Target 5 | Target 6 | Target 7 | Consume |
|--------|----------|----------|----------|----------|----------|----------|----------|---------|
| sensor 1 | 0.447    | 0.916    | 0.275    | 0.462    | 0.213    | 0.234    | 0.098    | 0.03    |
| sensor 2 | 0.439    | 0.861    | 0.543    | 0.471    | 0.26     | 0.547    | 0.606    | 0.03    |
| sensor 3 | 0.655    | 0.597    | 0.388    | 0.032    | 0.516    | 0.171    | 0.671    | 0.02    |
| sensor 4 | 0.01     | 0.231    | 0.075    | 0.164    | 0.016    | 0.036    | 0.713    | 0.04    |
| sensor 5 | 0.982    | 0.031    | 0.413    | 0.386    | 0.987    | 0.66     | 0.615    | 0.03    |
According to the weight calculation model, the importance of the target parameters is judged, and the target parameter interval threat judgment matrix is constructed, as shown in Table 3. Then the consistency test is carried out, and the random consistency index is found to be 1.12, and then \( CR^- = -0.227 \) and \( CR^+ = 0.390 \) are calculated. The mean value is \( CR = 0.081 < 0.1 \), and the consistency test is passed. According to formulas (10) and (11), \( \alpha = 1.13 \) and \( \beta = 0.86 \) are calculated.

According to the weighting step, \( \tilde{W}^+ \) and \( \tilde{W}^- \) can be obtained:

\[
\tilde{W}^+ = (0.049, 0.349, 0.169, 0.15, 0.142) \\
\tilde{W}^- = (0.079, 0.455, 0.259, 0.176, 0.161)
\]

(15)

Then calculate its average value, which is the weight of each target threat element, and the result is \( W = (0.064, 0.402, 0.214, 0.163, 0.1515) \).

Finally, according to the indicators of each element, the threat degrees of the seven targets can be obtained as follows:

\[
T = (0.6513, 0.7718, 0.7063, 0.5866, 0.8132, 0.8129, 0.6574)
\]

(16)

| sensor 6 | 0.172 | 0.197 | 0.97 | 0.987 | 0.072 | 0.8 | 0.127 | 0.02 |
|---------|-------|-------|------|-------|-------|-----|-------|-----|
| sensor 7 | 0.04  | 0.163 | 0.617| 0.242 | 0.803 | 0.971| 0.758 | 0.05 |
| sensor 8 | 0.277 | 0.094 | 0.047| 0.945 | 0.499 | 0.65 | 0.632 | 0.02 |
| sensor 9 | 0.044 | 0.081 | 0.707| 0.068 | 0.33  | 0.934| 0.158 | 0.03 |
| sensor 10| 0.057 | 0.353 | 0.197| 0.33  | 0.27  | 0.916| 0.043 | 0.02 |

| index | type | speed | height | route shortcut | distance |
|-------|------|-------|--------|---------------|----------|
| type  | [1,1]| [0.15,0.19]| [0.22,0.29]| [0.27,0.44]| [0.38,0.71]|
| speed | [5.4,6.6]| [1,1]| [1.25,2.75]| [1.25,3.75]| [2.4,4.6]|
| height| [3,4,4,6]| [0.36,0.8]| [1,1]| [1.16]| [1,16]|
| route shortcut | [2.25,3.75]| [0.27,0.8]| [0.625,1]| [1,1]| [0.4,1,6]|
| distance | [1.4,2.6]| [0.22,0.42]| [0.625,1]| [0.625,2.5]| [1,1]|

4.2. Comparison of results

The basic algorithm and improved algorithm are used to solve the objective function. The simulation parameters are set as follows: the number of populations is set to 60, the inertia weight is set to 0.8, the learning factors \( C_1 \) and \( C_2 \) are both set to 1.5, and the collision factors \( C_3 \) and \( C_4 \) are set to 1 and 4, the threshold is set to \( e_{op} = 0.01 \), \( e_{eg} = 0 \), 200 Monte Carlo experiments are carried out, the results are shown in Figure 1, and the alliance scheme is shown in Table 4.

**Figure 1.** Comparison before and after improvement.

It can be seen from Figure 1 that the algorithm can be stabilized before and after the improvement, and the solution of the multi-sensor detection alliance can be obtained. The basic particle swarm...
algorithm is stable after 25 iterations, with a fitness value of 9.511. The improved particle swarm algorithm reached stability after 13 iterations, with a fitness value of 10.63, which improved the convergence speed and fitness.

**Table 4.** Comparison of alliance allocation plans.

|   | forward |   | later |   |
|---|---------|---|-------|---|
|   | target | sensor | adaptability | consume | senso | adaptability | consume |
| 1 | 3.5,6,8 | 23.18 | 0.09 | 1.3,5 | 26.0563 | 0.08 |
| 2 | 1.2,3,9 | 21.5818 | 0.11 | 1 | 30.5167 | 0.03 |
| 3 | 5.6,9 | 26.1212 | 0.08 | 6.10 | 34.1675 | 0.04 |
| 4 | 2.6,8 | 34.3286 | 0.07 | 1.6,8 | 34.1929 | 0.07 |
| 5 | 5.7,8,10 | 23.8208 | 0.12 | 5.10 | 31.13 | 0.05 |
| 6 | 5.6,10 | 33.9429 | 0.07 | 10 | 45.785 | 0.02 |
| 7 | 2,3,4,8 | 26.22 | 0.11 | 3.5,8 | 27.4 | 0.07 |

4.3. Comparison and analysis of different algorithms for solving alliance schemes

The ant colony algorithm and the improved bee colony algorithm and the improved particle swarm algorithm are used to solve the problem. After 200 iterations, the comparison of the three algorithm results is shown in Figure 2.

It can be seen from Figure 2 that the three algorithms finally reached stability, and the ant colony algorithm reached stability after 42 iterations, with a fitness value of 8.634. The improved bee colony algorithm is stable after 31 iterations, with a fitness value of 9.063. The improved particle swarm algorithm reached stability in 17 iterations, with a fitness value of 10.66. It can be seen that the improved particle swarm algorithm has better solution quality and convergence speed than the ant colony algorithm and the improved bee colony algorithm in solving the multi-sensor alliance scheme.

5. Conclusion

This paper proposes a multi-sensor alliance scheme based on NC-PSO. The simulation analysis shows that in the process of NC-PSO solving the sensor detection alliance, the convergence speed and accuracy of the algorithm are improved compared with the previous improvement and the comparison algorithm, which proves the rapidity and effectiveness of the algorithm, and algorithm can quickly get a better multi-sensor alliance program.

References

[1] Itshak T , Aleksandar J , Shimon N , et al. A Modified Distributed Bees Algorithm for Multi-Sensor Task Allocation[J]. Sensors, 2018, 18(3):759.

[2] Sri V R , Sukumar M . DYNAMIC TASK ALLOCATION IN MULTI-HOP WIRELESS SENSOR NETWORK WITH MOBILITY. 2016.

[3] Sun Z , Liu Y , Tao L . Attack localization task allocation in wireless sensor networks based on multi-objective binary particle swarm optimization[J]. Journal of Network and Computer Applications, 2018:29-40.

[4] Qi B , Miao X , Miao H , et al. Tasks Scheduling of Actuators Algorithm Based on PSO-ACO for WSAN[J]. Journal of Communications, 2016, 11(12):1066-1072.

[5] Tian-Long L I , Zhang J C . Air-to-Ground Multi-target Attack Firepower Assignment Based on Fusion Algorithm[J]. Electronics Optics & Control, 2019.

[6] Guo W Z , Jin-Shu S U , Chen C Y , et al. Self-adapted task allocation algorithm with complicated coalition in wireless sensor network[J]. Journal on Communications, 2014.

[7] Zhu X . Optimization of multi-task nodes in wireless sensor networks based on discrete particle swarm algorithm[J]. Journal of Intelligent and Fuzzy Systems, 2021(22):1-12.

[8] Sarita Devi , Deepika Garg. Hybrid genetic and particle swarm algorithm: redundancy allocation problem[J]. International Journal of System Assurance Engineering and Management, 2019(prepublish):1-7.