Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

# A scheduling method based on a hybrid genetic particle swarm algorithm for multifunction phased array radar*

Hao-wei ZHANG[†‡1], Jun-wei XIE[1], Wen-long LU[2], Chuan SHENG[1], Bin-feng ZONG[3]
(*[1]Air and Missile Defense College, Air Force Engineering University, Xi'an 710051, China*)
(*[2]Troop 94225, Rizhao 276000, China*)
(*[3]Troop 94710, Wuxi 214000, China*)
[†]E-mail: zhw_xhzf@163.com

Received June 21, 2016; Revision accepted Sept. 25, 2016; Crosschecked Nov. 23, 2017

**Abstract:** A hybrid optimization approach combining a particle swarm algorithm, a genetic algorithm, and a heuristic interleaving algorithm is proposed for scheduling tasks in the multifunction phased array radar. By optimizing parameters using chaos theory, designing the dynamic inertia weight for the particle swarm algorithm as well as introducing crossover operation and mutation operation of the genetic algorithm, both the efficiency and exploration ability of the hybrid algorithm are improved. Under the frame of the intelligence algorithm, the heuristic interleaving scheduling algorithm is presented to further use the time resource of the task waiting duration. A large-scale simulation demonstrates that the proposed algorithm is more robust and efficient than existing algorithms.

**Key words:** Phased array radar; Scheduling; Particle swarm algorithm; Genetic algorithm; Pulse interleave
https://doi.org/10.1631/FITEE.1601358 **CLC number:** TP391

## 1 Introduction

The multifunction phased array radar (MFPAR) can play the roles of many traditional radar systems and execute many tasks such as space surveillance, multiple targets tracking, and communication simultaneously. These abilities depend on its highly efficient time resource allocation. Optimal time resource allocation to the tasks in MFPAR is the key to improving its efficiency and releasing its full potential.

The scheduling algorithm, which allocates the time resource to radar tasks, has aroused intensive interest recently. Studies have shown that this problem is NP-hard. At present, the solutions to the problem can be divided into two kinds, heuristic algorithms and intelligence algorithms. The heuristic algorithms schedule the tasks satisfying the preset rules. Orman *et al.* (1996) and Zeng *et al.* (2004a; 2004b) proposed the highest task mode priority first algorithm. They preset the task mode priorities for each kind of task and scheduled prior the task with the highest task mode priority. Butler (1998) and Reinoso-Rondinel *et al.* (2010) developed the time balance algorithm. They introduced the time balance value to indicate the task urgency, and scheduled prior the tasks with an imminent deadline. Huizing and Bloemen (1996) and Jiménez *et al.* (2009; 2012) proposed a queue-based algorithm. They divided the tasks into several queues by ordinations and the values of the task parameters, and used the earliest deadline first (EDF) algorithm or the first in first out (FIFO) algorithm to schedule the tasks in each queue. Lu *et al.* (2006; 2011; 2013) and Cheng *et al.*(2009b) mapped the task mode priority and the deadline on the same layer to calculate the task synthetic priority, and proposed the highest task mode priority and earliest

[‡] Corresponding author
[*] Project supported by the National Natural Science Foundation of China (Nos. 61503408 and 61601504)
ORCID: Hao-wei ZHANG, http://orcid.org/0000-0001-8786-6736

deadline first algorithm. Cheng *et al.* (2008) and Chen *et al.* (2011) considered further the scheduling principle of the timeliness based on Lu *et al.* (2006; 2011; 2013) and Cheng *et al.* (2009b), and put forward an algorithm based on gain. Mir and Abdelaziz (2012), and Mir and Guitouni (2014) proposed the notion of 'variable dwell time' for task scheduling, and adopted multi-nested heuristic algorithms to schedule as many tasks as possible. Galati *et al.* (2015a; 2015b) and Galati and Emilio (2015) studied the scheduling problem in MFPAR which is based on an active phased array of the conformal type, and presented an efficient heuristic method. The heuristic algorithms have low computation complexity but usually find suboptimal solutions. In addition, when the task scale is too large, the above solutions will be far from satisfactory.

Intelligence algorithms can always find better solutions because of their use of swarm search and continuous evolution. Zhou *et al.* (2006), Wang *et al.* (2014), and Zhang *et al.* (2016) used a genetic algorithm to solve the structured objective functions in MFPAR. de Jong and van Norden (2007) put forward a hybrid algorithm which combines fuzzy logic and the evolutionary algorithm to schedule MFPAR tasks. However, as far as we are aware, research on the intelligence algorithm for the scheduling problem in MFPAR is rather limited, and most traditional approaches use only a single optimization algorithm to solve the problem.

The genetic algorithm, particle swarm algorithm, and heuristic interleaving algorithm are combined in this study. An effective hybrid scheduling algorithm is proposed for MFPAR. First, the scheduling principles of importance, urgency, and timeliness in MFPAR are synthesized to structure the objective function under multiple resource constraints. Then, the hybrid algorithm is proposed to explore solutions for the objective function. Through optimizing the particle swarm algorithm by a chaos parameter, designing the dynamic inertia weight and introducing crossover and mutation operations of the genetic algorithm, the hybrid algorithm can achieve quick convergence and global exploration. The heuristic interleaving algorithm is presented to improve the time resource utility further in the intelligence algorithm's framework. In the end, the simulation results verified the effectiveness of the proposed algorithm.

## 2 Modeling

### 2.1 Radar task model

The overall MFPAR scheduling structure is shown in Fig. 1. When the target is captured by MFPAR, a number of radar tasks will be generated. The scheduling algorithm will analyze all the request tasks and the resource constraints to divide the tasks into three parts: execution tasks, delayed tasks, and deleted tasks. The delayed tasks will require to be executed later, and the deleted tasks will be abandoned. The tasks' request order is usually search→confirmation→tracking (→tracking loss→tracking maintenance). The tracking tasks can be divided into precise tracking, normal tracking, and monitoring.

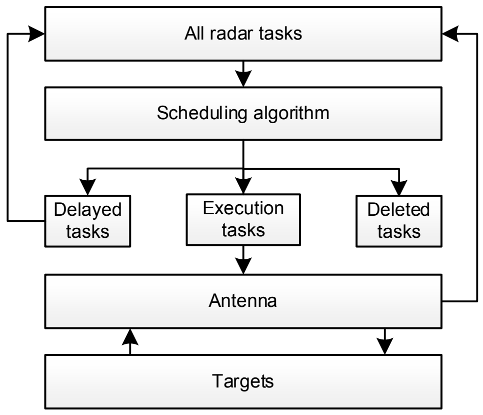

**Fig. 1 The overall MFPAR scheduling**

Fig. 2 shows the task model in MFPAR. The radar task contains mainly three parts: transmitting duration, waiting duration, and receiving duration. The number *i* task can be described as

$$T_i = \{P_i, t_{\mathrm{a}i}, t_{\mathrm{x}i}, t_{\mathrm{w}i}, t_{\mathrm{r}i}, P_{\mathrm{t}i}, t_{\mathrm{dw}i}, w_i, t_{\mathrm{d}i}, \Delta t_i, N_{\mathrm{um}i}\}, \quad (1)$$

where $P_i$ is the task priority, $t_{\mathrm{a}i}$ the task request time, $t_{\mathrm{x}i}$ the transmitting duration, $t_{\mathrm{w}i}$ the waiting duration (which depends on the target distance), $t_{\mathrm{r}i}$ the receiving duration, $P_{\mathrm{t}i}$ the power consumption of the task, $t_{\mathrm{dw}i}$ the task dwell time, $w_i$ the time-window, $t_{\mathrm{d}i}$ the task deadline, $\Delta t_i$ the sample interval between the tasks, and $N_{\mathrm{um}i}$ the task execution number. The dwell time satisfies

$$t_{\mathrm{dw}i} = t_{\mathrm{x}i} + t_{\mathrm{r}i}. \quad (2)$$

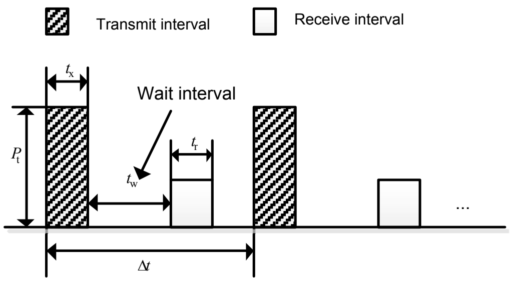

**Fig. 2 The task model in MFPAR**

Huizing and Bloemen (1996) proposed the notion 'time-window', which means the task execution time can shift within the window. The time-window makes it possible to schedule more tasks. After adding the time-window, a task deadline satisfies

$$t_{\mathrm{d}i} = t_{\mathrm{a}i} + w_i. \tag{3}$$

Each task should be executed before its deadline; otherwise it will be useless because of the target escape. The request time between tasks satisfies

$$t_{\mathrm{a}i} = t_{\mathrm{e}(i-1)} + \Delta t_i, \tag{4}$$

where $t_{\mathrm{e}(i-1)}$ is the execution time of the former task.

## 2.2 Resource constraint model

### 2.2.1 Time resource constraint

Scheduling in MFPAR is based on the scheduling interval (SI), when the radar must process the returning signals in the previous SI and determine the scheduling sequences in the next SI (Cheng *et al.*, 2009b). In an SI, all the tasks to be executed must satisfy

$$\sum_{i=1}^{N_1} t_{\mathrm{dw}i} \le t_{\mathrm{SI}}, \tag{5}$$

where $N_1$ is the total number of tasks to be executed, and $t_{\mathrm{SI}}$ is the SI duration. If a task cannot be executed in an SI, it will be delayed to latter SIs or be deleted. At the same time, the transmitting duration and the receiving duration in radar tasks are non-preempted. So, the $N_1$ tasks should also satisfy

$$\bigcap_{i_1=1}^{N_1}\left[(t_{\mathrm{e}i_1}, t_{\mathrm{e}i_1}+t_{\mathrm{x}i_1})\bigcup(t_{\mathrm{e}i_1}+t_{\mathrm{x}i_1}+t_{\mathrm{w}i_1}, t_{\mathrm{e}i_1}+t_{\mathrm{x}i_1}+t_{\mathrm{w}i_1}+t_{\mathrm{r}i_1})\right]=\varnothing. \tag{6}$$

### 2.2.2 Energy resource constraint

Due to its heat dissipation performance restriction, the radar transmitter must keep satisfying the transient constraint as (Ghosh *et al.*, 2004):

$$P_\tau(t) \le \overline{P}_{\tau\max}, \tag{7}$$

where $\overline{P}_{\tau\max}$ is the power threshold and $P_\tau(t)$ is the power consumed by the radar in time $t$. We can obtain

$$P_\tau(t) = \frac{1}{\tau}\int_0^t p(x)\exp[(x-t)/\tau]\mathrm{d}x, \tag{8}$$

where $p(x)$ is the instantaneous power dissipation in the radar, and $\tau$ is the look-back period. $\tau$ characterizes the heat dissipation performance of the radar. When the transient power consumption of the system reaches the power threshold, the radar must stop working or will be damaged.

## 2.3 Formulation of the objective

During MFPAR scheduling tasks, the following principles (Cheng *et al.*, 2008; Chen *et al.*, 2011) should be followed: (1) Importance principle, which means that the more important task should be prior scheduled; (2) urgency principle, which means that the more urgent task should be prior scheduled; (3) timeliness principle, which means that the task execution time should be close to its request time for the radar to adjust to the dynamic working situations. The first two principles reflect the task's inherent attributes, and the third one reflects the interactive attribute between the scheduling algorithm and the task. To reflect the paratactic relationship between the first two principles and the interactive relationship between the scheduling algorithm and the task, the objective function is structured as

$$\begin{aligned} &o(P, t_\mathrm{a}, w, t_\mathrm{start}, t_\mathrm{e}) \\ &= [o_1(P) + o_2(t_\mathrm{a}, w, t_\mathrm{start})]o_3(t_\mathrm{e}, t_\mathrm{a}, w), \end{aligned} \tag{9}$$

where $o_1(P)$ is the function of the task priority ($P$), and it represents the task importance. The higher $o_1(P)$ is, the more important the task is. $o_2(t_\mathrm{a}, w, t_\mathrm{start})$ is the function of the relative distance between the task deadline ($t_\mathrm{a}$+$w$) and the start time ($t_\mathrm{start}$) in SI, which describes the urgency of the task. The closer the task deadline ($t_a$+$w$) is to $t_{\text{start}}$, the more urgent the task is. $o_3(t_e, t_a, w)$ is the function of the relative distance between the task request time ($t_a$) and its execution time ($t_e$) in the time-window ($w$), and it describes the the timeliness of scheduling the task. The smaller the relative distance between $t_e$ and $t_a$ is, the greater timeliness of scheduling the task is. A reasonable objective function can be chosen as $o_1(P)=P$, $o_2(t_a, w, t_{\text{start}})=\exp[-2(t_a+w-t_{\text{start}})/t_{\text{SI}}]$, and $o_3(t_e, t_a, w)=[1-(t_e-t_a)/w]$. In the function, $o_1(P)$ is the increasing function of $P$, $o_2(t_a, w, t_{\text{start}})$ is the increasing function of the relative distance between the task deadline and the start time in SI. $o_3(t_e, t_a, w)$ is the increasing function of the relative distance between the task request time and its execution time in the time-window. The constants or the form in the objective function can be changed as long as it can appropriately reflect the relationship between three scheduling principles. It can be seen from Eq. (9) that the structured objective function integrates many scheduling principles, and the performance of the algorithm can be guaranteed in many aspects.

Assume that there are $N$ request tasks in an SI, and the numbers of executed, delayed, and deleted tasks are $N_1$, $N_2$, and $N_3$, respectively. Obviously, $N=N_1+N_2+N_3$. Thus, the optimal model of the task scheduling in MFPAR can be described as

$$\max \sum_{i}^{N} o(P, t_{\text{a}}, w, t_{\text{start}}, t_{\text{e}})$$

s.t.

$$\begin{cases} \max(t_{\text{a}i_1} - w_{i_1}, t_{\text{start}}) \le t_{\text{e}i_1}, & i_1 = 1,2,\ldots,N_1, \\ t_{\text{e}i_1} < \min(t_{\text{a}i_1} + w_{i_1}, t_{\text{start}} + t_{\text{SI}}), & i_1 = 1,2,\ldots,N_1, \\ \bigcap_{i_1=1}^{N_1} \left[ (t_{\text{e}i_1}, t_{\text{e}i_1} + t_{\text{x}i_1}) \bigcup (t_{\text{e}i_1} + t_{\text{x}i_1} + t_{\text{w}i_1}, t_{\text{e}i_1} + t_{\text{x}i_1} + t_{\text{w}i_1} + t_{\text{r}i_1}) \right] & \\ \quad = \varnothing, & \\ \sum_{i=1}^{N_1} t_{\text{dw}i_1} \le t_{\text{SI}}, P_\tau(t) \le \overline{P}_{\tau\max}, & \\ t_{\text{a}i_2} + w_{i_2} \ge t_{\text{s}} + t_{\text{SI}}, & i_2 = 1,2,\ldots,N_2, \\ t_{\text{a}i_3} + w_{i_3} < t_{\text{s}} + t_{\text{SI}}, & i_3 = 1,2,\ldots,N_3, \end{cases} \tag{10}$$

where the first four constraints are for the tasks to be executed, and the remaining two constraints are for the tasks to be delayed and to be deleted, respectively. Eq. (10) reflects the fact that the scheduling problem in MFPAR is an optimization one under multiple constraints, and an efficient algorithm should be used to solve the problem.

## 3 Hybrid genetic particle swarm algorithm

### 3.1 Particle swarm algorithm

The particle swarm algorithm has the advantages of quick convergence and simple use compared with other intelligence algorithms. Thus, it is widely applied in dealing with scheduling problems (Akhshabi *et al.*, 2014; Liu *et al.*, 2015; Tian *et al.*, 2016). The main idea of the algorithm is based on a natural phenomenon: to search for food, each bird in a flock will determine the velocity through its own experience and interactive information with the other members of the flock. In the particle swarm algorithm, the solutions to the objective function are regarded as individual particles. In the iteration number $t$, particle $k$ explores the optimal solution by updating its position and its velocity, through tracking the best solution $p_{\text{best}}(t)$ that the particle has achieved so far and the best solution $g_{\text{best}}(t)$ that the swarm has achieved so far:

$$\begin{aligned} v_k(t+1) = w \cdot v_k(t) + c_1 \cdot r_1 \cdot [p_{\text{best}}(t) - x_k(t)] \\ + c_2 \cdot r_2 \cdot [g_{\text{best}}(t) - x_k(t)], \end{aligned} \tag{11}$$

$$x_k(t+1) = x_k(t) + v_k(t), \tag{12}$$

where $c_1$ and $c_2$ are the study factors. They can regulate the maximum step when the particle is flying to $p_{\text{best}}(t)$ and $g_{\text{best}}(t)$, and they are usually adopted as $c_1=c_2=2$. $r_1$ and $r_2$ are random values belonging to (0, 1). $w$ is the inertia weight used to balance the particles' exploration ability between the whole solution region and local solution region.

From Eqs. (11) and (12), it can be seen that the algorithm will stop exploring solutions when all the particles reach $g_{\text{best}}(t)$. It is easy to stick the algorithm on the local optimum. Consequently, the particle swarm algorithm should be modified in application.

### 3.2 Chaos parameter optimization

The chaotic sequence has ergodicity and randomness, and can be used to optimize the swarm algorithms while exploring solutions. Thus, the logistic equation is adopted to generate the chaotic sequence (Ott *et al.*, 1990):

$$\lambda(t+1)=\mu\lambda(t)[1-\lambda(t)], \tag{13}$$

where $\lambda(t)$ is obtained after the chaos variable $\lambda$ has iterated $t$ times, and $\lambda\in[0, 1]$. $\mu$ is the parameter that controls the chaos state, and $\mu\in[0, 4]$. When $\mu=4$ and $\lambda\notin\{0.25, 0.5, 0.75\}$, the sequence will have a fully chaotic property and the trajectory of the chaos variable will spread throughout the solution region. Therefore, the updating parameters of the particle velocities can be optimized by the chaos theory as

$$r_l(t+1)=4r_l(t)[1-r_l(t)],\ r_l(t)\in(0,1),\ l=1,2, \tag{14}$$

where $r_1$ and $r_2$ will obtain ergodicity and randomness from Eq. (14), and it guarantees that the algorithm can explore solutions in the whole solution region and find the global optimum with a high probability.

## 3.3 Dynamic inertia weight

In the particle swarm algorithm, the inertia weight $w$ determines the algorithm exploration ability in the whole solution region and the local solution region. The larger $w$ endows the algorithm with a stronger exploration ability in the whole solution region, and the smaller $w$ gives the algorithm better exploration performance in the local solution region. Therefore, the larger $w$ should be adopted to locate the optimum range at the beginning early exploration stage, and the smaller $w$ should be adopted to find the optimum accurately in the later exploration stage. The dynamic $w$ is designed as

$$w(t)=(w_{\max}-w_{\min})t_{\max}/t+w_{\min}, \tag{15}$$

where $w_{\max}$ and $w_{\min}$ are the maximum and minimum of the inertia weight, respectively. $t_{\max}$ and $t$ are the maximum and the current number of iterations, separately. It can be seen from Eq. (15) that the declining linear inertia weight will improve the exploration efficiency of the algorithm.

## 3.4 Crossover and mutation operations

To break out the stagnation when all the particles reach $g_{\text{best}}(t)$ of the swarm, the crossover and mutation operations in the genetic algorithm are introduced to change the positions and the velocities of the partial particles. The two operations can destroy the stability of the particle swarm and drive the particles to explore the better solutions in the whole solution region.

The genetic algorithm simulates the heredity and the evolution procedure of the organism in nature. In the genetic algorithm, the selection, crossover, and mutation operations are the three main steps to reserve the fittest solutions and ensure that the algorithm can find the optimum. The selected individuals are called ‘parents’. The individuals generated through the crossover operation or mutation operation are called ‘children’. Here, the following crossover and mutation operators are designed to enhance the variability of the swarm, which can drive the particles to explore better solutions.

### 3.4.1 Crossover operation

Assume that the swarm scale of the particle is $N_{\text{pop}}$ and that the selection probability of the crossover operation is $P_{\text{c}}$. After sorting the fitness values (Eq. (9)) of the particles diminishingly, the first $P_{\text{c}}N_{\text{pop}}$ particles are selected as parents. For each parent $m$, a different parent $n$ will be chosen to apply crossover with it, and the positions and the velocities of the generated child are calculated as

$$x'_m(t)=P_{\text{b}}x_m(t)+(1-P_{\text{b}})x_n(t), \tag{16}$$

$$v'_m(t)=\left[v_m(t)+v_n(t)\right]\left|v_m(t)\right|\Big/\sqrt{v_m^2(t)+v_n^2(t)}, \tag{17}$$

where $P_{\text{b}}$ is the random value that belongs to (0, 1). If the fitness value of the child is better than that of parent $m$, the position and velocity of the child generated will be reserved. Otherwise, the position and velocity of the parent will be reserved. Through Eqs. (16) and (17), the child inherits the information from both of the elite parent particles. They ensure the child generated to be in the feasible solution region. What is more, along with the change of $P_{\text{b}}$, the algorithm can explore further in the solution region between the two extrema. The two equations enhance the exploration possibility of the global optimum of the algorithm.

### 3.4.2 Mutation operation

Assume that the mutation probability of all the particles is $P_{\text{m}}$, and the positions of the children generated from the selected parents of $P_m N_{pop}$ can be calculated as

$$x''_m(t) = x_m(t)(1+P_b). \tag{18}$$

In the mutation operation, the velocity of the child is equal to that of the parent. If the fitness value of the child is better than that of the corresponding parent, the position and velocity of the child generated will be reserved. Otherwise, that of the parent will be reserved. The child generated from Eq. (18) is in the neighbor of the parent. Similarly to Eq. (16), Eq. (18) enhances the possibility of finding the global optimum of the algorithm.

### 3.5 Heuristic interleaving algorithm

It can be seen from Eq. (10) that the scheduling tasks in MFPAR should satisfy multiple constraints. In traditional scheduling algorithms (Zeng *et al.*, 2004a; 2004b; Lu *et al.*, 2006; 2011; 2013; Mir and Abdelaziz, 2012), the whole radar dwell is regarded as non-preempted. The pulse interleaving technique makes it possible to execute the transmitting duration or receiving duration of other tasks during the task waiting duration, and it improves the time utility. However, it complicates the scheduling analysis. The two pulse interleaving ways are shown in Fig. 3. It can be seen that the interleaved two tasks should satisfy the following time constraint:

$$\begin{cases} t_{w1} \ge t_{x2}, \\ t_{r1} \le t_{w2}, \\ t_{x2} + t_{w2} \ge t_{w1} + t_{r1}, \end{cases} \tag{19a}$$

$$t_{w1} \ge t_{x2} + t_{w2} + t_{r2}. \tag{19b}$$

The two tasks should also satisfy the energy constraint in Eq. (7). According to Mir and Guitouni (2014), the energy constraint is ignored to simplify the application of the pulse interleaving technique. According to Cheng *et al.* (2009a; 2009b), the specific task interleaving ways as Eq. (19) should be considered to the application of the pulse interleaving. They are not yet satisfied. Thus, a heuristic interleaving algorithm is put forward as follows, and it contains mainly two parts, time constraint analysis and energy constraint analysis.

In an SI, the remaining time line is initialized as $[t_{start}, t_{end}]$, and the power pointer is $P_{t0}$. $t_{start}$ and $t_{end}$ are the start time and the end time of SI, respectively. The number of request tasks is $N$, and the number components in the particle is also $N$. A component represents the candidate execution time of each of request task. First, sort the components by the FIFO principle, and sign them as tasks 1, 2, 3,…, $N$. Then, analyze the time constraint to the transmitting duration of task 1:

$$\begin{cases} t_{a1} \ge t_{start}, \\ t_{a1} + t_{x1} \le t_{end}. \end{cases} \tag{20}$$

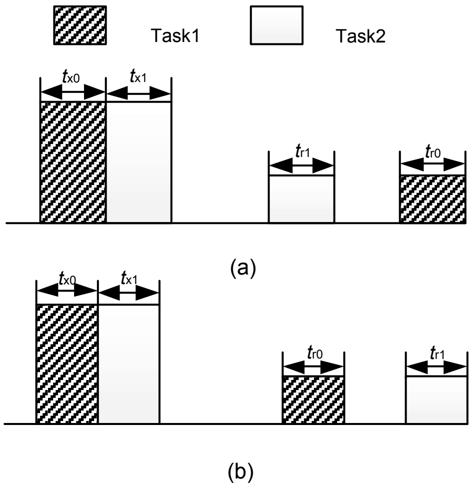

**Fig. 3 The two pulse interleaving ways in MFPAR: (a) pulse interleaving way 1; (b) pulse interleaving way 2**

If it does not satisfy the time constraint, set the task to the delayed or deleted queue according to Eq. (10). Otherwise, update the remaining time line as $[t_{start}, t_{a1}]$, $[t_{a1}+t_{x1}, t_{end}]$, and analyze the time constraint to the receiving duration of task 1

$$\begin{cases} t_{a1} + t_{x1} + t_{w1} \ge t_{a1} + t_{x1}, \\ t_{a1} + t_{x1} + t_{w1} + t_{r1} \le t_{end}. \end{cases} \tag{21}$$

When it does not satisfy the time constraint, set the task to the delayed or deleted queue according to Eq. (10), and reset the remaining time line as $[t_{start}, t_{end}]$. If it satisfies the time constraint, continue to test whether task 1 satisfies the energy constraint:

$$P_{t0} e^{-t_{x1}/\tau} + P_{t1}(1 - e^{-t_{x1}/\tau}) \le \overline{P}_{\tau\max}. \tag{22}$$

If task 1 does not satisfy the energy constraint, set the task to the delayed or deleted queue according to Eq. (10), and reset the remaining time line as [$t_{\text{start}}$, $t_{\text{end}}$]. If task 1 also satisfies the energy constraint, update the remaining time line as [$t_{\text{start}}$, $t_{\text{a1}}$], [$t_{\text{a1}}+t_{\text{x1}}$, $t_{\text{a1}}+t_{\text{x1}}+t_{\text{w1}}$], [$t_{\text{a1}}+t_{\text{x1}}+t_{\text{w1}}+t_{\text{r1}}$, $t_{\text{end}}$], and update the power pointer $P_{\text{t0}}$ as

$$P_{\text{t0}} = P_{\text{t0}}\mathrm{e}^{-t_{\text{x1}}/\tau} + P_{\text{t1}}(1-\mathrm{e}^{-t_{\text{x1}}/\tau}). \tag{23}$$

Then analyze the time and energy resource constraints to the remaining tasks sequentially. The heuristic interleaving algorithm can be summarized as: (1) test the task transmitting duration and receiving duration by the left and right boundaries of the time line, and test the energy constraint to the task; (2) update the time line and power pointer. Note that with the update of the time line, the time line will be separated into many pieces. When analyzing the time constraint for a duration, the time line pieces should all be tested. In the following tasks, the start time of a duration may be less than the left boundary of one time line piece. So, the first constraint in Eq. (21) is necessary. In this way, the specific task interleaving ways can be ignored and the interleaving analysis can be simplified. In addition, the fitness value of the particle can be calculated quickly.

### 3.6 Procedure of the hybrid algorithm

The overall procedure of the hybrid algorithm is shown in Fig. 4.

Step 1: parameter initialization. Set the swarm scale as $N_{\text{pop}}$, the maximum number of iterations as $t_{\max}$, the maximum inertia weight as $w_{\max}$, the minimum inertia weight as $w_{\min}$, the probability of crossover and that of mutation as $P_{\text{c}}$ and $P_{\text{m}}$ respectively. Calculate $r_1$, $r_2$, and $w(t)$ according to Eqs. (13)–(15).

Step 2: particle swarm initialization. For all the $N$ request tasks in an SI, generate the $N_{\text{pop}}$ candidate scheduling sequence (particles). Each sequence contains $N$ components standing for the candidate execution times of the request tasks. The execution times satisfy the first constraint in Eq. (10). The velocities of the particles are generated randomly. The velocities satisfy $v_k \in (-1,1)$.

Step 3: calculating the fitness values. Analyze the interleaving ways to the candidate sequences according to Eqs. (20)–(23), and calculate the fitness values (Eq. (9)) of all the particles in the swarm based on Eq. (10). Then, obtain $p_{\text{best}}(t)$ and $g_{\text{best}}(t)$ in $t$ iterations.

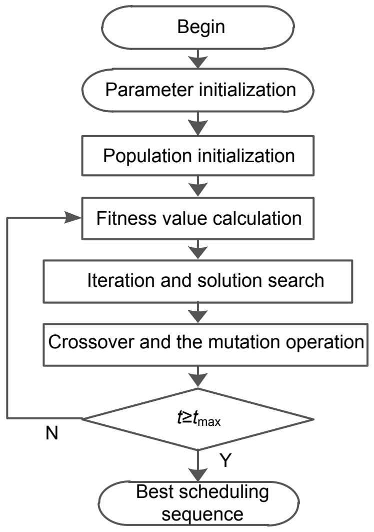
Begin
Parameter initialization
Population initialization
Fitness value calculation
Iteration and solution search
Crossover and the mutation operation
$t \geq t_{\max}$
N
Y
Best scheduling sequence

**Fig. 4 The flowchart of the hybrid algorithm**

Step 4: iteration and exploration of the solutions. Iterate the particle swarm algorithm to explore the solutions to the objective function according to Eqs. (11) and (12), and then update $p_{\text{best}}(t)$ and $g_{\text{best}}(t)$.

Step 5: crossover and mutation operations. The crossover and mutation operations are carried out to the particles according to Eqs. (16)–(18), which can enhance the variability of the particle swarm and drive the swarm to explore better solutions further. Then, update $p_{\text{best}}(t)$ and $g_{\text{best}}(t)$.

Step 6: when the iteration number $t \geq t_{\max}$, the algorithm stops. Output the best candidate scheduling sequence according to the particle that has $g_{\text{best}}(t)$. Otherwise, return to step 3.

## 4 Simulation and analysis

### 4.1 Performance evaluation indexes

According to radar scheduling principles, these indexes are chosen to evaluate the performance of the algorithms:

1. The successful scheduling ratio (SSR), which is the ratio between the number of all successfully

scheduled tasks and that of all request tasks, can be expressed as

$$S_{\mathrm{SR}} = N_{\mathrm{suc}} / N_{\mathrm{total}}, \tag{24}$$

where $N_{\mathrm{suc}}$ and $N_{\mathrm{total}}$ are the number of all successfully scheduled tasks and that of all request tasks, respectively. SSR reflects the urgency principle of the algorithm. Under the finite time resource constraint, the more tasks that can be scheduled successfully, the better the algorithm performs.

2. The time utilization ratio (TUR), which is defined as the ratio between the time consumption of the successfully scheduled tasks and the totally available time resource, can be expressed as

$$T_{\mathrm{UR}} = \sum_{i=1}^{N_{\mathrm{suc}}} \frac{t_{\mathrm{x}i} + t_{\mathrm{r}i}}{T_{\mathrm{total}}}, \tag{25}$$

where $T_{\mathrm{total}}$ is the total time resource. When scheduling tasks, the algorithms should use the time resource efficiently to schedule as many tasks as possible.

3. The high value ratio (HVR), which is the ratio between the total priority of all successfully scheduled tasks and that in all request tasks. HVR can reflect whether the algorithm has prior scheduled more important tasks. It can be expressed as

$$H_{\mathrm{VR}} = \sum_{i=1}^{N_{\mathrm{suc}}} P_i \Big/ \sum_{i=1}^{N_{\mathrm{total}}} P_i. \quad (26$$

4. The average time shift ratio (ATSR), which is defined as the shift degree between the execution time of the successfully scheduled tasks and their request time, can reflect whether the algorithm satisfies the timeliness scheduling principle. It can be expressed as

$$A_{\mathrm{TSR}} = \frac{1}{N_{\mathrm{suc}}} \sum_{i=1}^{N_{\mathrm{suc}}} \frac{|t_{\mathrm{a}i} - t_{\mathrm{e}i}|}{w_i}. \tag{27}$$

Note that a lower ATSR means a better search and tracking performance to MFPAR.

## 4.2 Simulation parameters

The whole simulation framework is based on Kuo *et al.* (2005). The number of targets ($N_{\mathrm{t}}$) increases from 10 to 100, which denotes the different workload of MFPAR. The objective function is chosen as $o_1(P)=P$, $o_2(t_{\mathrm{a}},w,t_{\mathrm{start}})=\exp[-2(t_{\mathrm{a}}+w-t_{\mathrm{start}})/t_{\mathrm{SI}}]$, SI ($t_{\mathrm{SI}}$) is set as 50 ms, $\bar{P}_{\tau\max}$=1.25 kW, and $\tau$=200. and $o_3(t_{\mathrm{e}}, t_{\mathrm{a}}, w)=1-(t_{\mathrm{e}}-t_{\mathrm{a}})/w$. In the hybrid genetic particle swarm algorithm, $N_{\mathrm{pop}}$=100, $w_{\max}$=0.9, $w_{\min}$=0.2, $P_{\mathrm{c}}$=0.5, $P_{\mathrm{m}}$=0.1, and $t_{\max}$=200. The hybrid genetic algorithm (Zhou *et al.*, 2006) (HGA) and the two heuristic scheduling algorithms are used as the baseline in the simulations. The heuristic algorithms are referenced from Lu *et al.* (2013) (heuristic o/interleaving, i.e., heuristic without interleaving) and Cheng *et al.* (2009b) (heuristic w/interleaving, i.e., heuristic with interleaving) separately. In HGA, set $N_{\mathrm{pop}}$=100, $t_{\max}$=500, $P_{\mathrm{c}}$=0.5, and $P_{\mathrm{m}}$=0.1. The task parameters are shown in Table 1.

## 4.3 Simulation results and analysis

The statistical results are shown in Figs. 5a–5d and Table 2 after 50 simulations.

Figs. 5a and 5b compare the SSR and HVR of the four algorithms. The two heuristic algorithms and their HVRs begin to decrease when the target number is over 10. By contrast, the proposed algorithm starts to drop tasks when the target number is over 40 and its HVR starts to decline. The maximum workload of MFPAR is significantly improved. In addition, SSR of the heuristic algorithm with interleaving catches up with that of HGA when the target number is 40. HVR of the heuristic algorithm with interleaving exceeds that of HGA when the target number is 60. The two points show the lack of robustness in HGA, especially when faced with large-scale radar tasks. However, both SSR and HVR decrease the speed of the proposed algorithm which is much slower than the other three algorithms after dropping tasks. It displays the global exploration and robustness of the proposed algorithm.

Fig. 5c shows the TUR comparison of the four algorithms. TURs in the three other algorithms are much higher than that of the heuristic algorithm without interleaving. The main cause is that the pulse interleaving technique can make full use of the waiting durations in radar tasks, and it ensures more tasks to be scheduled successfully. Though the pulse interleaving technique is also adopted in the heuristic algorithm with interleaving and HGA, they find only local optimum. However, the proposed algorithm

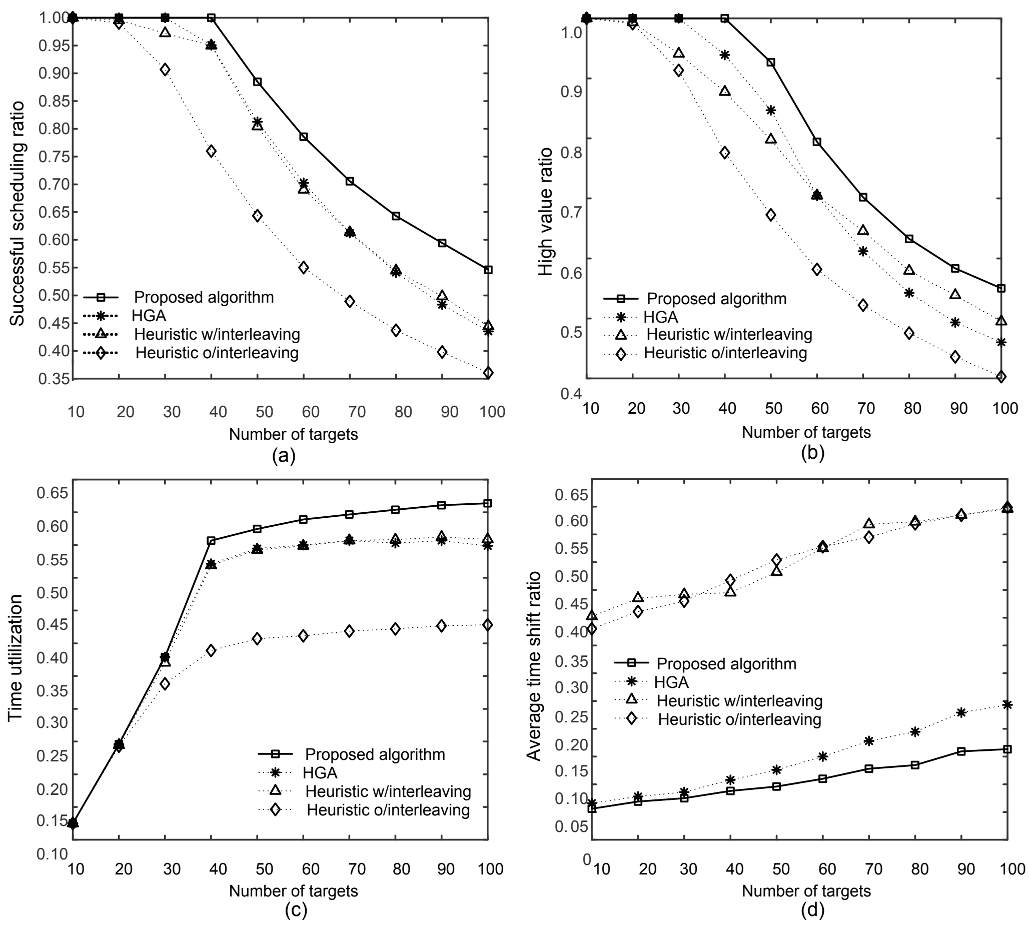

**Fig. 5 Statistical results: (a) comparison of the successful scheduling ratio; (b)comparison of the high value ratio; (c) comparison of the time utilization ratio; (d) Comparison of the average time shift ratio**

**Table 1 Parameters of tasks**

| Task | Task priority | Parameters of dwell ($t_x$, $t_w$, $t_r$) | Power (kW) | Time-window (ms) | Sample interval (ms) |
|---|---|---|---|---|---|
| Confirmation | 6 | (1, DV, 1) | 5 | 20 | 150 |
| Precise track | 5 | (0.5, DV, 0.5) | 4 | 20 | 100–200 |
| Track loss | 4 | (1, DV, 1) | 5 | 30 | UV |
| Normal track | 3 | (0.5, DV, 0.5) | 3 | 100 | 250-500 |
| Monitor | 2 | (0.5, DV, 0.5) | 3 | 200 | 1000 |
| Search | 1 | (1, DV, 1) | 5 | NV | 10 |

DV represents ‘dynamic value’. NV represents ‘no value’, i.e., the task has no time constraint. UV represents ‘unfixed value’, which means that the task ‘track loss’ has no fixed sample duration

**Table 2 Runtime of the algorithms at different $N_t$**

| Algorithm | Runtime (ms) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 1020 | 1030 | 1040 | 1050 | 1060 | 1070 | 1080 | 1090 | 1100 |
| Proposed | 18.6 | 22.2 | 24.6 | 25.8 | 29.0 | 31.6 | 33.4 | 35.5 | 40.2 | 41.6 |
| HGA | 138.5 | 142.4 | 148.1 | 152.3 | 163.4 | 178.8 | 200.4 | 202.3 | 221.5 | 245.4 |

finds the global optimum and obtains the highest TUR among the four algorithms.

Fig. 5d compares the ATSRs of the four algorithms. From this we can see that the ATSRs of the intelligence algorithms (HGA and the proposed algorithm) are much lower than that of the heuristic algorithms. This is because the heuristic algorithms always prior schedule tasks with the highest synthetic priority in the available time. However, in the proposed algorithm and HGA, the swarm evolutionary method ensures better solutions to the objective function. In addition, the proposed algorithm obtains a lower ATSR than HGA.

Table 2 compares the runtime of the proposed algorithm and HGA. It can be seen that the runtime of the proposed algorithm is less than 50 ms and it satisfies the real-time demand of the system. However, the runtime of HGA exceeds 50 ms and does not satisfy the real-time demand. This is because several optimization methods such as chaos parameter optimization and the design of the dynamic inertia weight are adopted in the proposed algorithm. Thus, it is more efficient than HGA.

Above all, the proposed algorithm can not only find better solutions than the three algorithms, but also explore the solutions more quickly than HGA. What matters most is that the proposed algorithm improves SSR by 53%, HVR by 38%, and TUR by 47%. It decreases ATSR by 73% compared with the heuristic algorithm without interleaving.

## 5 Conclusions

Realizing optimal task scheduling is the key to unlocking the full potential of MFPAR. Aiming at this problem, the optimal objective function was established and a hybrid algorithm was proposed. The objective function integrated multiple scheduling principles. It ensures better algorithm performance in many aspects. The proposed algorithm composited the particle swarm algorithm, genetic algorithms and the heuristic interleaving algorithm, and many optimization methods were introduced. The simulations show that the proposed algorithm possesses the merits of global exploration, fast convergence, and robustness to solve the scheduling problem in MFPAR. By employing the hybrid algorithm, MFPAR can be used more efficiently, and radar tasks can be allocated more appropriately.

## References

Akhshabi, M., Tavakkoli-Moghaddam, R., Rahnamay-Roodposhti, F., 2014. A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. *Int. J. Adv. Manuf. Technol.*, **70**(5-8):1181-1188. https://doi.org/10.1007/s00170-013-5351-9

Butler, J.M., 1998. Tracking and Control in Multi-function Radar. PhD Thesis, UCL University of London, London.

Chen, J., Tian, Z., Wang, L., *et al.*, 2011. Adaptive simultaneous multi-beam dwell scheduling algorithm for multifunction phased array radars. *J. Inform. Comput. Sci.*, **8**(14):3051-3061.

Cheng, T., He, Z.S., Tang, T., 2008. Dwell scheduling algorithm for multifunction phased array radars based on the scheduling gain. *J. Syst. Eng. Electron.*, **19**(3):479-485. https://doi.org/10.1016/S1004-4132(08)60110-3

Cheng, T., He, Z.S., Li, H.Y., 2009a. Adaptive dwell scheduling for digital array radar based on online pulse interleaving. *Chin. J. Electron.*, **18**(3):574-578.

Cheng, T., He, Z.S., Tang, T., 2009b. Novel radar dwell scheduling algorithm based on pulse interleaving. *J. Syst. Eng. Electron.*, **20**(2):247-253.

de Jong, J.L., van Norden, W.L., 2007. Application of hybrid metaheuristics in sensor management. *Aerosp. Sci. Technol.*, **11**(4):295-302. https://doi.org/10.1016/j.ast.2006.09.001

Galati, G., Emilio, G.P., 2015. Scheduling methods for a conformal, phased array multifunction radar. Proc. 2nd Int. Conf. on Advances in Information Processing and Communication Technology, p.103-108.

Galati, G., Madia, F., Carta, P., *et al.*, 2015a. Time for a change in phased array radar architectures–Part I: planar vs. conformal arrays. Proc. 16th Int. Radar Symp., p.912-917. https://doi.org/10.1109/IRS.2015.7226275

Galati, G., Madia, F., Carta, P., *et al.*, 2015b. Time for a change in phased array radar architectures–Part II: the d-Radar. Proc. Int. Radar Symp., p.24-26. https://doi.org/10.1109/IRS.2015.7226276

Ghosh, S., Hansen, J., Rajkumar, R., *et al.*, 2004. Integrated resource management and scheduling with multi-resource constraints. Proc. 25th IEEE Int. Real-Time Systems Symp., p.12-22. https://doi.org/10.1109/REAL.2004.25

Huizing, A.G., Bloemen, A.A.F., 1996. An efficient scheduling algorithm for a multifunction radar. IEEE Int. Symp. on Phased Array Systems and Technology, p.359-364. https://doi.org/10.1109/PAST.1996.566115

Jiménez, M.I., Izquierdo, A., Villacorta, J.J., *et al.*, 2009. Analysis and design of multifunction radar task schedulers based on queue. Proc. 28th Digital Avionics Systems Conf., p.295-302.

https://doi.org/10.1109/DASC.2009.5347448

Jiménez, M.I., Val, L.D., Villacorta, J.J., *et al*., 2012. Design of task scheduling process for a multifunction radar. *IET Radar Sonar Navig*., **6**(5):341-347. https://doi.org/10.1049/iet-rsn.2011.0309

Kuo, T.W., Chao, Y.S., Kuo, C.F., *et al.*, 2005. Real-time dwell scheduling of component-oriented phased array radars. *IEEE Trans*. *Comput*., **54**(1):47-60. https://doi.org/10.1109/TC.2005.10

Liu, L.L., Hu, R.S., Hu, X.P., *et al*., 2015. A hybrid PSO-GA algorithm for job shop scheduling in machine tool production. *Int*. *J*. *Prod*. *Res*., **53**(19):5755-5781. https://doi.org/10.1080/00207543.2014.994714

Lu, J.B., Hu, W.D., Yu, W.X., 2006. Study on real-time task scheduling of multifunction phased array radars. *Acta Electron*. *Sin*., **34**(4):732-736 (in Chinese). https://doi.org/10.3321/j.issn:0372-2112.2006.04.032

Lu, J.B., Xiao, H., Xi, Z.M., *et al*, 2011. Multifunction phased array radar resource management: real-time scheduling algorithm. *J*. *Comput*. *Inform*. *Syst*., **7**(2):385-393.

Lu, J.B., Xiao, H., Xi, Z.M., *et al*, 2013. Phased array radar resource management: task scheduling and performance evaluation. *J*. *Comput*. *Inform*. *Syst*., **9**(3):1131-1138.

Mir, H.S., Abdelaziz, F.B., 2012. Cyclic task scheduling for multifunction radar. *IEEE Trans*. *Autom*. *Sci*. *Eng*., **9**(3): 529-537. https://doi.org/10.1109/TASE.2012.2197857

Mir, H.S., Guitouni, A., 2014. Variable dwell time task scheduling for multifunction radar. *IEEE Trans*. *Autom*. *Sci*. *Eng*., **11**(2):463-472. https://doi.org/10.1109/TASE.2013.2285014

Orman, A.J., Potts, C.N., Shahani, A.K., *et al*., 1996. Scheduling for a multifunction phased array radar system. *Eur*. *J*. *Oper*. *Res*., **90**(1):13-25 https://doi.org/10.1016/0377-2217(95)00307-X

Ott, E., Grebogi, C., Yorke, J.A., 1990. Controlling chaos. *Phys*. *Rev*. *Lett*., **64**(11):1196-1199. https://doi.org/10.1103/PhysRevLett.64.1196

Reinoso-Rondinel, R., Yu, T.Y., Torres, S., 2010. Multifunction phased-array radar: time balance scheduler for adaptive weather sensing. *J*. *Atmos*. *Ocean*. *Technol*., **27**(11): 1854-1867. https://doi.org/10.1175/2010JTECHA1420.1

Tian, G.D., Ren, Y.P., Zhou, M.C., 2016. Dual-objective scheduling of rescue vehicles to distinguish forest fires via differential evolution and particle swarm optimization combined algorithm. *IEEE Trans*. *Intell*. *Transp*. *Syst*., **17**(11):3009-3021. https://doi.org/10.1109/TITS.2015.2505323

Wang, S.J., He, J., Wang, B., *et al.*, 2014. Research on adaptive scheduling algorithm based on improved genetic algorithm for multifunctional phased array radar. Int. Conf. on Future Computer and Communication Engineering, p.24-28. https://doi.org/10.2991/icfcce-14.2014.7

Zeng, G., Lu, J.B., Hu, W.D., 2004a. Research on adaptive scheduling algorithm for multifunction Akhshabi, M., radar. *Mod*. *Radar*, **26**(6):14-18 (in Chinese). https://doi.org/10.16592/j.cnki.1004-7859.2004.06.006

Zeng, G., Hu, W.D., Lu, J.B., *et al*., 2004b. The simulation on adaptive scheduling for multifunction phased array radars. *J*. *Syst*. *Simul*., **16**(9):2026-2029 (in Chinese). https://doi.org/10.16182/j.cnki.joss.2004.09.044

Zhang, H.W., Xie, J.W., Sheng, C., 2016. Scheduling method for phased array radar over chaos adaptively genetic algorithm. Proc. 6th Int. Conf. on Information Science and Technology, p.111-116. https://doi.org/10.1109/ICIST.2016.7483395

Zhou, Y., Wang, G.Y., Wang, X.S., *et al.*, 2006. Optimal scheduling using hybrid GA with heuristic rules for phased array radar. *Syst*. *Eng*. *Electron*., **28**(7):992-996, 1005 (in Chinese). https://doi.org/10.3321/j.issn:1001-506X.2006.07.014