The Journal of Systems & Software 205 (2023) 111801

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

A hybrid grey wolf optimizer using opposition-based learning, sine cosine algorithm and reinforcement learning for reliable scheduling and resource allocation☆

Man Zhao, Rui Hou, Hui Li *, Min Ren

School of Computer, China University of Geosciences, Wuhan, China

ARTICLE INFO

Article history:
Received 25 January 2023
Received in revised form 25 May 2023
Accepted 3 July 2023
Available online 7 July 2023

Keywords:
Grey wolf optimizer
Reinforcement learning
Optimized scheduling
Resource allocation
Space debris tracking

ABSTRACT

As the number of space debris in geosynchronous Earth orbits continues to grow, the threat posed by space debris to satellites surveillance is increasing, and the available orbital resources are also decreasing. Thus, reasonably scheduling and allocating the resources for space object tracking has become vital. This paper establishes an optimization model for the resource allocation and scheduling problem for space debris tracking. A fusion algorithm that combines the grey wolf optimizer, opposition-based learning, sine cosine search strategy, and reinforcement learning was proposed and used to solve the problem. Six groups of realistic data were selected based on the relevant background information of space debris tracking to test the validity and effectiveness of the proposed algorithm. The performance of the state-of-the-art optimization algorithms was compared with that of the proposed algorithms. The result of the experiment indicates that the proposed algorithm effectively solves the resource allocation and scheduling problem for space debris tracking.

1. Introduction

Space debris refers to all artificial non-functional objects, including their fragments, located in Earth orbit or re-entering the atmosphere. The number of space debris monitored by space tracking and surveillance devices in many countries has risen dramatically in recent years (Allworth et al., 2021). Since space debris has severely endangered aerospace safety, effectively allocating ground- and space-based observation equipment resources and improving the scheduling system's overall performance has become one of the key research fields.

Zheng et al. (2016) used Pareto and particle swarm optimization (PSO) to solve the problem of resource scheduling through grouped dynamic matrix programming. Liu et al. (2020) designed an index to evaluate the degree of conflict in the usage of ground station resources and used a heuristic algorithm and Lagrange decomposition algorithm to solve the resource scheduling problem. Zhou et al. (2019) designed an intelligent resource scheduling model with reinforcement learning based on user satisfaction for multi-satellite and Cui and Wang (2008) proposed a grid scheduling algorithm based on resource monitoring and load adjustment. There have been few studies on resource scheduling algorithms for space debris tracking based on swarm intelligence. Existing studies mainly focus on swarm intelligence algorithms' task planning and resource scheduling. This paper combines the grey wolf optimizer (GWO) and reinforcement learning, and proposes an improved GWO based on reinforcement learning (QGWO) and applied to the resource optimization scheduling for space debris tracking. The experimental results show that the improved Grey Wolf algorithm mentioned in this paper is more suitable for the observation of spatial debris due to its faster convergence, fewer parameters and ability to find optimal solutions compared to other optimization algorithms.

The following sections of the paper are arranged as follows. Section 2 introduces the preliminaries of the research. Sections 3 and 4 present the proposed algorithmic approach and solution for tracking resource scheduling and optimization respectively. The experimental design and analysis of the result are explained in Section 5. Finally, the paper is concluded and potential future directions are provided in Section 6.

2. Preliminaries

2.1. Grey wolf optimizer (GWO)

The GWO (Mirjalili et al., 2014; Chaudhari, 2022) is a new type of swarm intelligence optimization algorithm inspired by the hunting behaviours of a grey wolf pack. It is characterized by a simple structure, minimal parameters, and easy implementation.

☆ Editor: Prof W. Eric Wong.

* Corresponding author.
E-mail addresses: zhaoman@cug.edu.cn (M. Zhao), hourui@cug.edu.cn (R. Hou), lihui@cug.edu.cn (H. Li), renmin@cug.edu.cn (M. Ren).

https://doi.org/10.1016/j.jss.2023.111801
0164-1212/

It has a good performance in terms of problem-solving accuracy and convergence speed.

2.2. Improved GWO

2.2.1. Opposition-based learning

In many cases, learning starts from a random point. It starts from scratch and intends to move towards an existing solution. If one is searching for x and believes it is beneficial to search in the opposite direction, then the first step is to calculate the opposite $\tilde{x}$.

The pseudocode for the GWO is shown in Algorithm1:

Algorithm 1. Grey wolf optimizer

Initialize
Grey wolf population X_i(i=1, 2, …, n)
Initialize
a, A, and C
Calculate the fitness value of each grey wolf
X_α= Best individual
X_β= Second best individual
X_δ= Third best individual
while (t<maximum number of iterations)
for each individual
Update the position of the grey wolf
end for
Update a, A, and C
Calculate the fitness value of all individuals
Update X_ω, X_β and X_δ
t=t+1
end while
Return X_α

The flow chart of the GWO is shown in Fig. 1:

Let $x \in \aleph$ be a real number defined in a certain interval $x \in [a, b]$. The corresponding number x is defined as follows:

$$\tilde{x} = a + b - x \tag{1}$$

For a = 0 and b = 1, there is:

$$\tilde{x} = 1 - x \tag{2}$$

Similarly, the opposite number can be defined in the multidimensional case.

Let $P(x_1, x_2 \ldots, x_n)$ be a point $(x_1, x_2 \ldots, x_n \in \aleph)$ in the n-dimensional coordinate system and $x_i \in [a_i, b_i]$. The opposite point $\tilde{P}$ is entirely defined by the coordinates $\tilde{x}_1, \ldots \tilde{x}_n$, and:

$$\tilde{x}_i = a_i + b_i - x_i \quad i = 1, \ldots, n \tag{3}$$

Let $f(x)$ be the function in focus, and $g(\cdot)$ be the appropriate evaluation function. If $x \in [a, b]$ is the initial (random) guess and $\tilde{x}$ is its opposite value, then $f(x)$ and $f(\tilde{x})$ are calculated in each iteration. If $g(f(x)) \geq g(f(\tilde{x}))$, then continue use x to learn.

The evaluation function $g(\cdot)$ is used as a measure of optimality to compare the applicability of the results, such as fitness function, reward/punishment function, and error function. Considering interval [a1, b1] in Fig. 2, the solution to a given problem can be found by repeatedly verifying guesses and opposite guesses. The opposite number xo of the initial guess x is generated. Depending on which estimate or counter estimate is closer to the solution, the search interval can be halved recursively until the estimate or counter estimate is close enough to the existing solution.

According to the opposition-based learning (OBL) proposed by Tizhoosh (2006), the current wolf and its opposite wolf have been considered to show signs of improvement to guess the current solution. It gives a superior chance that the opposite solution is closer to the global optimal solution than any solution. Each solution Y_i has a unique opposite solution Y_{opi}. Solution $OP(Y_1, Y_2 \ldots, Y_n)$ is calculated according to the following formula.

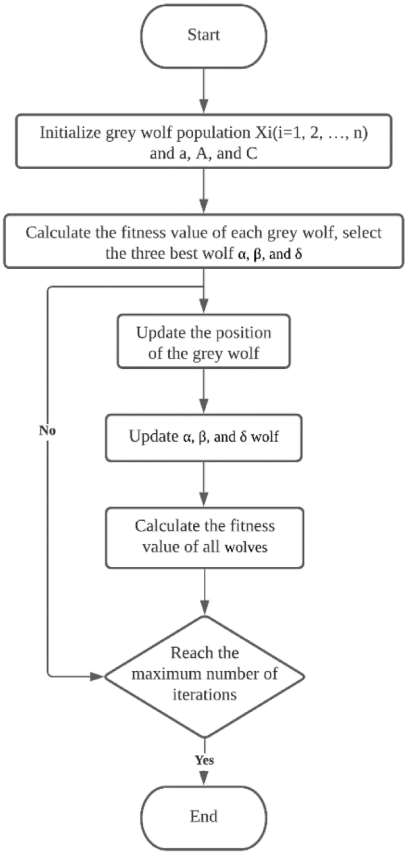

Fig. 1. Flow chart for the grey wolf optimizer.

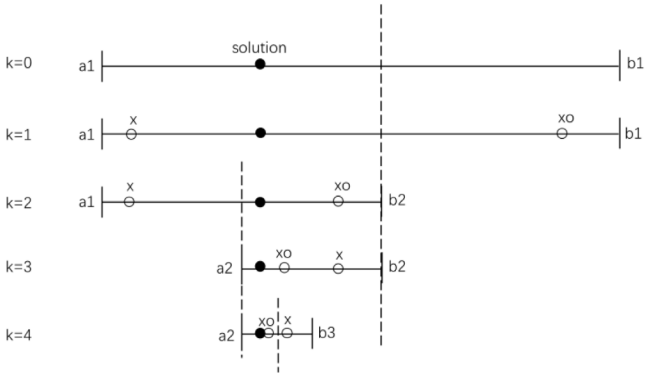

Fig. 2. Opposition-based learning diagram.

$$Y'_{ij} = a_i + b_i - Y_i, \ i \in 1, 2, \ldots, n \tag{4}$$

2.2.2. Sine cosine search strategy

The sine cosine algorithm (SCA) is a new algorithm proposed by Mirjalili (2016). It can generate multiple random initial candidate solutions in the algorithm and make them move toward the optimal solution (Paikray et al., 2021). Using random parameters and adaptive parameters to locate the current solution can efficiently converge to the global optimal solution (Xu and Long, 2018).

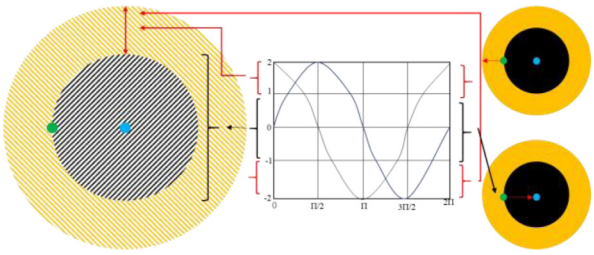

Fig. 3. The optimization process of sine cosine algorithm. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The search solution process of the SCA has only two steps. The first is exploration, which finds the feasible solutions among all random solutions, and the second is exploitation. The rate of change in the random solution is lower than the rate of the exploration process (Lawal et al., 2021). The equation is as follows:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 * \sin(r_2) * \left| r_3 P_i^t - X_i^t \right|, r_4 < 0.5 \\ X_i^t + r_1 * \cos(r_2) * \left| r_3 P_i^t - X_i^t \right|, r_4 > 0.5 \end{cases} \tag{5}$$

where X_i^t denotes the position of the tth generation of the current individual in the ith dimension, r_2 denotes a random number between 0 to 2π, r_3 denotes a random number between 0 to 2, r_4 denotes a random number between 0 to 1, and P_i^t denotes the position of the optimal individual position variable during the tth iteration in the ith dimension.

$$r_1 = a - t\frac{a}{T} \tag{6}$$

where a denotes a constant, t denotes the current number of iterations, and T denotes the maximum number of iterations.

The optimization process of SCA is shown in Fig. 3. If the return values of the sine and cosine functions are between -1 and 1 (indicated by the light grey area in the figure), the candidate solution can be used to fully search the local search space. If the return values of the sine and cosine functions are greater than 1 or less than -1 (indicated by the light yellow area in the figure), the candidate solution can be used to fully search the global space.

The SCA proposed by Mirjalili (2016) is introduced in the GWO. After the GWO's position update is completed, a sine cosine search is performed. The sine cosine search is mathematically expressed as follows:

$$X(t+1) = \begin{cases} X(t) + r_3 \times sin(r_4) \times D^\theta, < 0.5 \\ X(t) + r_3 \times cos(r_4) \times D^\theta, r_5 \geq 0.5 \end{cases} \tag{7}$$

$$D^\theta = \{|r_6 X_i^\alpha (t-1) - X_i(t)|| \; i = 1, 2, \ldots, d\} \tag{8}$$

$$r_3 = \mathrm{a} - \mathrm{a} \times \frac{t}{maxit}, r_4 \in \left(0, 360^\circ\right) \tag{9}$$

where r_3 decreases adaptively with the increase of the number of iterations, and a denotes a constant with a value of 2. Moreover, r_5 and r_6 are random numbers uniformly distributed on [0,1].

2.2.3. Reinforcement learning (RL)

Reinforcement learning is a search process that explores and evaluates at the same time. The agent chooses an action to perform on the environment. After the environment is affected, its state will change, and a corresponding reinforcement signal will be generated. The reinforcement signal contains two aspects:

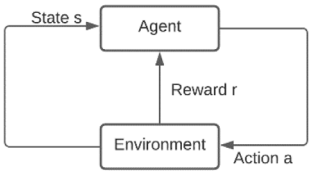

Fig. 4. Reinforcement learning diagram.

reward and punishment, which then provide the agent feedback, which determines the next action according to the current feedback signal. Its principle is to increase the probability of receiving the feedback signal. The basic principles of reinforcement learning are shown in Fig. 4.

A typical reinforcement learning problem formula consists of the state space S and a set of actions A in each state (Pardeshi et al., 2022). The agent takes action a from A when in state s and is moved by the environment to state s' while receiving the reward $R_a\left(t, s'\right)$. The goal is to choose the appropriate action in each state to maximize the cumulative reward. The mapping from the current state s to action a is called a strategy (Emary et al., 2018).

Through long-term self-games, the neural network can well evaluate the state values and strategies of large-scale problems. For example, AlphaGo Zero (Fu, 2019) regard training artificial chess players as a reinforcement learning problem. The idea is that the deep residual network outputs both the strategy and the state value simultaneously. A reliable search method is used to determine the parameters of the neural network.

Over time, many major breakthroughs have been made in reinforcement learning. Reinforcement learning can be divided into two categories: policy-based methods and value-based methods (Du and Ding, 2020). Q-learning algorithm is a typical example of value-based methods. In the learning process, the agent estimates the optimal strategy using the action with the highest expected q value (Mishra and Arora, 2022). The Q table is dynamically updated according to the reward, and the calculation is as follows:

$$\mathrm{Q}_{t+1}(s_t, a_t) \leftarrow Q(s_t, a_t) + \lambda \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{10}$$

The pseudocode of the Q-learning algorithm is as follows:

Algorithm 2. Q-learning algorithm

Initialization
Set state s and action a
for each state s_i, and action a_i
Set $Q(s_i, a_i)$=0
end for
Randomly select an initial state s,
While the termination condition is not met, **do**
Select the best action a from the action a_t in the current state s_t and Q table, and then get instant rewards
Find the new state s_{t+1},
Get the corresponding maximum Q value s_{t+1}
Update Q table according to Equation (10)
Update state $s_t \leftarrow s_{t+1}$
End While

3. Q-learning improved GWO

In the GWO, it has two actions for search (exploration and exploitation). The balancing between exploration and exploitation

controls the search behaviour, with over-exploration allowing the optimization process to converge quickly, but potentially leading to premature maturity. Over-exploitation increases the probability of finding a globally optimal solution, but usually at a lower rate of convergence thereby slowing down the process. The search action (exploration and exploitation) is selected by the parameter $\vec{A}_i$. When the random value of $\vec{A}_i$ is [−1, 1], the wolf population moves evenly and slowly with a smaller step size. This is a partial search process in which $|\vec{A}_i| < 1$ is forced to search globally. However, the uniform searching behaviour does not guarantee the global minimum value. Independent search action is a better indicator to ensure ultimate success, and the embedding of reinforcement learning can accomplish this task well. Therefore, the QGWO is proposed to obtain the optimal parameter $ExpRat(a)$ to improve the optimization performance of the GWO.

$$ExpRat = 2 - (t) * (2/T) \tag{11}$$

where t denotes the current iteration, T denotes the total number of iterations, and $ExpRat$ denotes the rate of exploration. Although this formula has been proved effective in solving many optimization problems, it still has the following disadvantages:

(1) In the later stage of algorithm iteration, it tends to fall into the local optimal solution since its exploration capability becomes limited.

(2) At the beginning of the algorithm, the optimizer has a great exploration capability. However, as the exploration capability reinforces, it may leave the promising area to the hopeless area.

(3) For all search agents, the exploration rate is the same, forcing the entire population to search in the same way.

To solve these disadvantages, it is necessary to find a way for adaptive exploration, and each search agent should be modelled separately. A mechanism to map the different states of the agent to an action set is needed to optimize the long-term goal or fitness function. Therefore, the following components of the system are defined:

(1) State–action mapping: The mapping between states and actions is usually nonlinear, and any nonlinear mapping model, such as a neural network model, can be used for modelling.

(2) Action set: To adapt to the exploration rate of each wolf, the following series of actions is taken: (a) increase the exploration rate, (b) decrease the exploration rate, and (c) maintain the exploration rate.

These three actions directly affect the exploration rate of the next iteration, as shown below:

$$ExpRat_i^{t+1} = \begin{cases} ExpRat_i^t * (1 + \Delta)(Increase\ action) \\ ExpRat_i^t * (1 - \Delta)(Decrease\ action) \\ ExpRat_i^t(Keep\ action) \end{cases} \tag{12}$$

where ExpRat_i^t denotes the exploration rate of the agent i at time t, and Δ denotes the deviation coefficient.

(3) Agent state: The executed action changes the state of the search agent, which is repositioned in the search space to obtain a new fitness value. The fitness history of a given agent is used to control its next action, so that the agent can accumulate and exploit its past decisions. Formally, the state increases, decreases, or maintains the fitness value of a search agent in the past continuous time span of T rounds:

$$\begin{aligned} State_t^i &= \sum_{t=1}^{T} sign\left(f_t^i - f_{t-1}^i\right) \\ &= [\ldots + sign\left(f_{t-3}^i - f_{t-4}^i\right) \\ &+ sign\left(f_{t-2}^i - f_{t-3}^i\right) \\ &+ sign\left(f_{t-1}^i - f_{t-2}^i\right) + sign\left(f_t^i - f_{t-1}^i\right)] \end{aligned} \tag{13}$$

where $State_t^i$ denotes the state of the given agent i at time t, $f\ (t)^i$ denotes the fitness function value of agent i at time t, and the symbol $sign\ (x)$ is defined as:

$$sign(x) = \begin{cases} 1, if\ x < 0 \\ -1, if\ x > 0 \\ 0, otherwise \end{cases} \tag{14}$$

The four states are set as follows:

$$\mathrm{State} = \begin{cases} exploration +,\ if\ \mathrm{State}_t^i \geq 0.9t \\ exploration -,\ 0 < if\ \mathrm{State}_t^i < 0.9t \\ exploitation -,\ if\ \mathrm{State}_t^i \geq 0 \\ exploitation +,\ if\ \mathrm{State}_t^i < 0 \end{cases} \tag{15}$$

They are exploration+, exploration −, exploitation +, and exploitation −. According to the value of State_t^i, the type of the current state is determined, and whether to increase the intensity of exploration or exploitation is decided.

(4) Feedback: When an agent leaves an area with a good fitness to one with a poor fitness, it receives a negative feedback. Conversely, when a search agent leaves a bad area for a better one, it receives a positive feedback. The agent's fitness value is an indicator of the surrounding search area, which determines whether the agent should obtain a positive or negative feedback. The feedback can be expressed as follows:

$$Feedback_i^t = \begin{cases} +1, iff\ \left(agent_i^{t+1}\right) < f\left(agent_i^t\right) \\ -1, iff\ \left(agent_i^{t+1}\right) \geq f\left(agent_i^t\right) \end{cases} \tag{16}$$

where $Feedback_i^t$ denotes the feedback of agent i at time t, and $f\left(agent_i^t\right)$ denotes the fitness value of agent i at time t.

In summary, the initial Q value is shown in Table 1:

The Q value table is updated according to Eq. (15). The pseudo code for the QGWO algorithm is as follows:

Algorithm 3. QGWO algorithm

Input: The number of grey wolves (n), the maximum number of iterations (Max_{Iter}), the state vector (T), and the exploration factor Δ.
Result: The best position and the fitness value of the wolf.
1) Randomly initialize n individual(s) (grey wolf).
2) **Initialize** A set of n values a_i represents the exploration rate of each wolf.
3) Find α, β, and δ based on the fitness value of the best three groups.
4) $t = 0$
while $t \leq Max_{Iter}$ **do**
For each $Wolf_i \in pack$ **do**
- Update the current position of $Wolf_i$
- Calculate the $Feedback_i$ according to Equation (16).
- Update the Q table according to Equation (10).
- Update the state vector of $Wolf_i$ according to Equation (15).
- Update the exploration parameter a_i of $Wolf_i$ according to Equation (12)
end
- Update A and C.
- Evaluate the fitness value of each individual.
- Update the positions of α, β, and δ.
$t = t + 1$
end
5) Choose the position of the optimal grey wolf

The flow chart for the algorithm is shown in Fig. 5.

Table 1
Initial Q value table.

State	Action		
	Increase action	Decrease action	Keep action
Exploration+	0	0	0
Exploration−	0	0	0
Exploitation−	0	0	0
Exploitation+	0	0	0

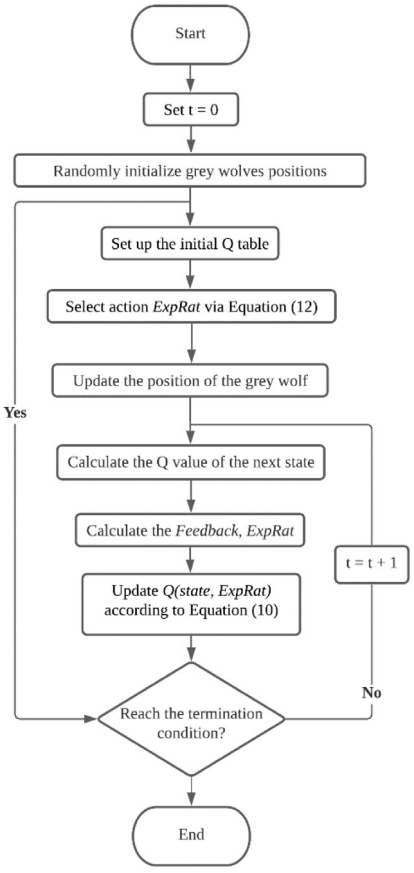

Fig. 5. Flow chart of QGWO.

4. Improved GWO based on reinforcement learning for tracking resource scheduling optimization

4.1. Coding process design

The solution vector structure of this model is spliced by combining the equipment selection and time selection of the tracking task. The input to the tracking task is a four-tuple: $\{Req_i, Tar_i, Dev_i, Rad_i\}$. Req_i denotes the tracking requirement, which is defined as a chromosome in this model. Tar_i represents the tracking target; a requirement contains multiple tracking targets, and the number of the tracking targets is the length of the chromosome. Dev_i represents the device, and a tracking task needs to call a tracking device. Rad_i represents the tracking radian, and one device corresponds to multiple tracking radians. Therefore, Dev_i and Rad_i jointly form the real number code of a locus, using two-dimensional encoding.

Each tracking target should be arranged with ground-based devices, and the tracking radian should be arranged for the load. Therefore, each tracking target should be arranged with two flag bits to allocate tasks. $Tar_i = 0$ denoting that the task Tar_i is not tracked, and a two-tuple is formed as a flag bit.

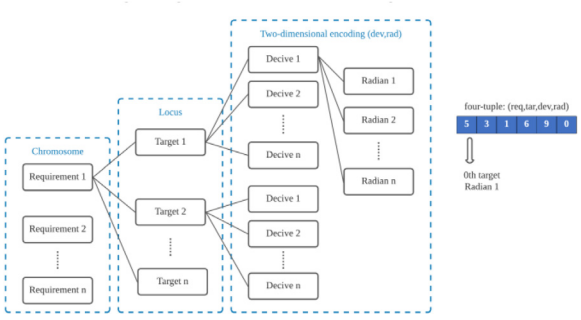

Fig. 6. Coding diagram.

Table 2
Parameters and descriptions.

Symbol	Descriptions
S	Number of tasks completed
s_i	Task i
Pri	Total priority
u_i	Priority of Task i
F_i	Top priority
C_{max}	Total economic cost
c_i	The total economic cost of Task i
P	Time penalty
M	Tracking target
H_m	Resource collection required for load modem
m	Load mode index, $m \in \{1, \ldots, M_i\}$
M_i	Tracking load mode used to complete the task
$T_{i,j}^{bov}$	Tracking task end time
$T_{i,j}^{eov}$	Tracking task start time
$\tau_{i,j}^{b}$	Planning end time
$\tau_{i,j}^{e}$	Planning start time
$\left[T_j^{bou}, T_j^{eou}\right]$	Optical device sunlight exposure time
y_i^t	Whether Task i is executed from time (0/1)
x_{im}	Whether Task i is executed in modem (0/1)
$E_{i,j}$	Measurement type j of tracking Task i
$Type_i^{data}$	Execution period of tracking Task i
$\left[T_m^b, T_m^e\right]$	Tracking condition interval

In this paper, the binary flag bits are mapped onto the unary flag bits for simplification. The specific rules are as follows:

(1) The first bit of the unary flag bit is set to 0, which means that no tracking is performed.

(2) $Tar_i = 0$ is defined as no tracking and is not repeatedly counted in the unary flag bit.

(3) The ascending sort of t is combined with the ascending sort of w one by one, which corresponds to the unary flag bit a.

(4) The mapping from (Dev_i, Rad_i) to Tar_i is saved.

The coding diagram is shown in Fig. 6:

4.2. Definition of basic variables

The basic variables are defined as shown in Table 2.

4.3. Evaluation function

Not only does tracking resource optimization scheduling involve many resources and tasks, but there are also complicated constraint relationships between them.

The evaluation values are established as follows:

(1) Define the time function

$$T = \frac{t_{l(e)} - t_{f(s)}}{t_{max}} \tag{17}$$

where $t_{l(e)}$ and $t_{f(s)}$ denote the time to end the last task and the time to start the first task, respectively, t_{max} denotes the maximum planning duration of the task, and T denotes the ratio of the completion time: The smaller the T, the faster the completion.

(2) Define the number of tasks completed function

$$S = \frac{\sum_{1\le i\le n} \frac{s_i}{s_{num}}}{n} \tag{18}$$

where s_i denotes the number of tasks completed by Task i, s_{num} represents the total number of tasks for Task i, n denotes the number of tasks, and S denotes the ratio of the number of tasks completed to the total number of tasks: the greater the S, the more tasks completed.

(3) Define the priority function

$$Pri = \frac{\sum_{1\le i\le n} \left(\frac{u_i}{F_i}\right)}{n} \tag{19}$$

where Pri denotes the ratio of the priority of the scheduled tasks to that of all tasks. The greater the Pri, the higher priority of the scheduled task.

$$C = \frac{\sum_{i=1}^{n} \sum_{m=1}^{M_i} \sum_{k\in H_m} c_k x_{im}}{c_{max}} \tag{20}$$

(4) Define the economic cost function

where C denotes the proportion of the economic resource consumed by the device to complete this batch of tasks in the total consumable resources. The greater the C, the more resources are consumed.

(5) If resource task conflict occurs, the punishment mechanism is added. If the task resources are occupied and the tracking scheme cannot be generated, the time penalty P is added to ensure the quality of the solution.

Based on the above five aspects, in this paper, the target function of the task planning model is as follows:

$$f(x) = \alpha(1-T) + \beta S + \gamma S + \delta(1-C) - P \tag{21}$$

where $\alpha, \beta, \gamma,$ and δ are the weight coefficients. In this model, before the initial setting, the importance of the four evaluation indicators is the same at 0.25.

4.4. Constraint setting

$$T_{i,j}^{\text{bov}} \le \tau_{i,j}^{b} < \tau_{i,j}^{e} \le T_{i,j}^{\text{eov}} \tag{22}$$

$$\left[\tau_{i,j}^{b}, \tau_{i,j}^{e}\right]_{\text{opt}} \subseteq \left(\left[T_j^{\text{bou}}, T_j^{\text{eou}}\right]_{\text{opt}} \cap \left[T_i^{\text{boi}}, T_i^{\text{eoi}}\right]\right) \tag{23}$$

$$\sum_{i=1}^{n} \sum_{m=1}^{M_i} r_{imk} x_{im} \le E_k, \forall k \in R \tag{24}$$

$$\sum_{t=0}^{T} y_i^t \le 1, i = 1, \ldots, n \tag{25}$$

$$\sum_{t\in T} t\left(y_i^t - y_j^t\right) \ge 0, u_i \ge u_j \tag{26}$$

$$\sum_{m\in M} x_{im} = \sum_{t\in T} y_i^t, i = 1, \ldots, n \tag{27}$$

$$E_{i,j} \ge E_m^{min}, \tau_{i,j}^{e} - \tau_{i,j}^{b} \ge WL_m^{min}, TS_{i,j} \le TS_m^{min} \tag{28}$$

$$\left\{Type_1^{data}, Type_2^{data}, \ldots, Type_{k_1}^{data}\right\}_{m,i} \subseteq \left\{Type_1^{data}, Type_2^{data}, \ldots, Type_{k_2}^{data}\right\} \tag{29}$$

$$\left[\tau_{i,j}^{b}, \tau_{i,j}^{e}\right] \subseteq \left[T_m^{b}, T_m^{e}\right] \tag{30}$$

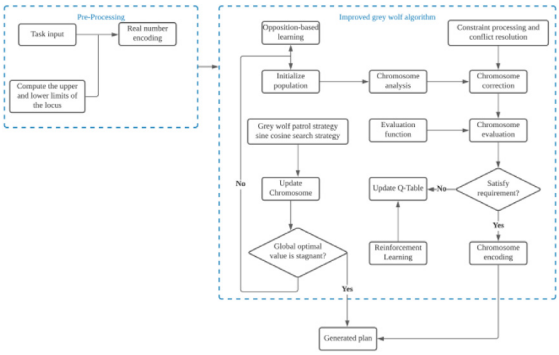

Fig. 7. Flow chart for the improved algorithm.

Eq. (22) represents that the start and end time of all tasks are finished within the tracking period. Eq. (23) represents the visibility constraint, that is, the telescopic device must be in the shadow of Earth, and the space target must be under the sunlight during optical tracking. Eq. (24) represents that the total amount of resources consumed by any load does not exceed the upper limit of resources. Eq. (25) represents that the start time of each task is unique. Eq. (26) represents that the priority of all tasks satisfies the constraint. Eq. (27) represents that all tasks are carried out using only one tracking load mode. Eq. (28) represents the tracking condition type of the tracking task. Eq. (29) represents the measurement type constraint of the tracking task, and Eq. (30) represents the execution period constraint of the tracking task.

4.5. Algorithm process

In this paper, based on the mathematical model of tracking resource optimization scheduling, a swarm intelligence algorithm improved on the basis of reinforcement learning was proposed to solve the problem, and the specific process is shown in Fig. 7.

First, after the tracking task planning system obtains the tracking resource data, according to the number of requirements and the number of tasks, the upper and lower limits of the locus are calculated, and then the real number is encoded. When initializing the pack, the OBL-improved strategy is used to perform OBL on the chromosome encoded by a real number to find a suboptimal solution for the initial generation. Subsequently, the chromosome is analysed and corrected. The correction is mainly for constraint processing and conflict resolution for the chromosome to make the planning results more realistic. The chromosome evaluation is to conduct a comprehensive evaluation of each chromosome in the pack, find the optimal chromosome, and keep it for the next iteration. The chromosome should be decoded to generate a plan if the planning result meets the planning requirement or the iteration is completed. If not so, the process enters the second improved strategy, Q-learning of reinforcement learning, to update the Q table and find the better optimization parameter a. Then the chromosome is updated. In this step, the third improved strategy, the sine cosine search strategy, is used in combination with the patrol strategy of the GWO for neighbourhood search. In the search process, whether the global optimal value is stagnant is decided. If it is stagnant, a plan is generated; otherwise, the pack is re-initialized and it enters the next iteration.

5. Experiment result and analysis

5.1. Environment setting

Table 3 shows the experimental environment information in this paper.

Table 3
Experimental environment information.

Testing platform	Windows
Hardware environment	Intel Core i5-10500@ 3.10 GHz
Software environment	IDEA 2020

Table 4
Data design table.

Task No.	Number of tasks	Number of ground stations involved in planning
Group 1	20	10
Group 2	50	10
Group 3	80	15
Group 4	100	20
Group 5	120	20
Group 6	150	20

Table 5
Parameters of PSO.

Parameter	Value
Population size	200
Max_{Iter}	200
Autonomous learning factor	2
Social learning factor	2
Maximum speed	2

5.2. Test set

To test the applicability and effectiveness of the algorithm for tracking resource optimization scheduling, six groups of data were constructed in the experiment according to the different numbers of tasks and space debris targets. The number of tasks was designed to be 10, 50, 80, 100, 120, and 150. The number of ground-based tracking devices was 10 or 20. The task planning for tracking resource optimization scheduling was performed, and the results were comprehensively analysed and compared. See Table 4 for specific settings.

5.3. Algorithm parameter setting

The parameter design of the optimized algorithm directly affects the result of task planning for tracking resource optimization scheduling. In this paper, many experiments were conducted on the design and value of parameters. Through the analysis and comparison of the planning result under different parameter values, the parameters used in the optimization algorithms in this paper were determined, as shown in Tables 5 and 6.

The same population size and Max_{Iter} are selected for a variety of algorithms. Researchers have experimentally shown that a suboptimal value for the learning factor of PSO can be obtained when the autonomous and social learning factors are constant, which is usually set to 2. In addition, since the locus has a high upper limit, the maximum speed is selected as 2. Also, the algorithm parameter λ in Q-learning is the amount of weighing results of the previous and current learning. A low λ causes the agent to follow the previous schemes too much, without getting new results. Usually, $\lambda = 0.5$ is set to balance old and new knowledge. Moreover, γ denotes the learning factor, and a value of 0.9 is usually used for better learning and selection. If it is adjusted to a smaller value, the reward at the endpoint will not spread correctly, and the agent may not be able to learn a strategy to reach the endpoint.

5.4. Comparison of algorithm operation efficiency

The OBL-based GWO was named OGWO. the GWO improved by the sine cosine search strategy was called SCGWO and the GWO improved by Q-learning was called QGWO. The results of the six data sets tested as shown in Table 4 were compared and analysed according to Eq. (21) and the combined evaluation value was calculated.

Table 6
Parameters of improved GWO.

Parameter	Value
Population size	200
Max_{Iter}	200
Q-learning- λ	0.5
Q-learning-γ	0.9
Optimization parameter a	Dynamically adjust by Q-learning

Table 7
Results of Group 1 algorithm before and after improvement.

Task No.	Planning algorithm	Comprehensive evaluation value	Planning time (s)
Group1	PSO	0.9	8
	GWO	0.95	9
	OGWO	1.0	8
	SCGWO	0.9	6
	QGWO	1.0	10

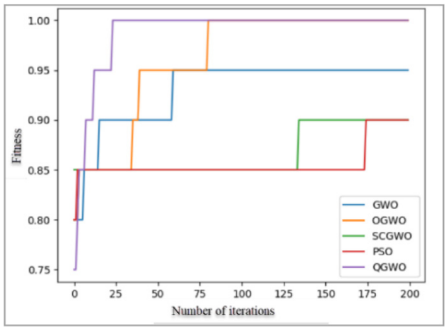

Fig. 8. Comparison of the algorithm fitness value of Group 1.

(1) Results and analysis of Group 1

As shown in Table 4 Group 1 and Table 7, the numbers of tasks completed by the OGWO and the QGWO are the greatest, and the planning time taken by the QGWO is the longest. In the case of low algorithm dimensionality, the optimal value can be reached in around 200 generations. The QGWO can quickly optimize in the first 25 generations, and the OGWO converges in 75 generations. The GWO converges in 60 generations, but the result is poor. In addition, the results of PSO and the SCGWO are relatively worse. Meanwhile, the initial fitness value of the OGWO in the first generation is higher than those of other algorithms. The line charts of the fitness iteration and the comparison of planning time are shown in Figs. 8 and 9.

(2) Results and analysis of Group 2

As shown in Table 4 Group 2 and Table 8, the QGWO has the highest comprehensive evaluation value, and the QGWO consumes more time. The convergence results calculated in the 200th generation surpass those of other algorithms, and there are more optimization times and frequent changes. The change curve is similar to that in Experiment 1: QGWO > OGWO > GWO > SCGWO > PSO. In the case of 50 dimensions, the performance of the QGWO is still outstanding, while that of PSO and SCGWO is still poor. The line chart for the fitness iteration and the comparison of planning time are shown in Figs. 10 and 11.

(3) Results and analysis of Group 3

As shown in Table 4 Group 3 and Table 9, the QGWO has the highest comprehensive evaluation value, and the QGWO takes the

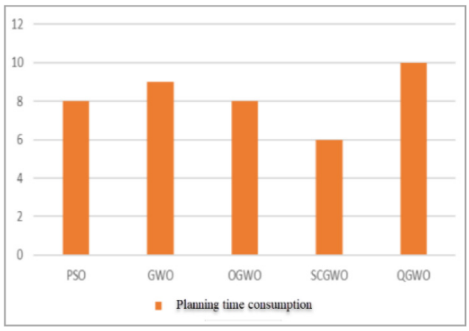

Fig. 9. Comparison of the time consumption of Group 1.

Table 8
Results of Group 2 algorithm before and after improvement.

Task No.	Planning algorithm	Comprehensive evaluation value	Planning time (s)
Group2	PSO	0.39	58
	GWO	0.41	67
	OGWO	0.42	78
	SCGWO	0.39	58
	QGWO	0.46	195

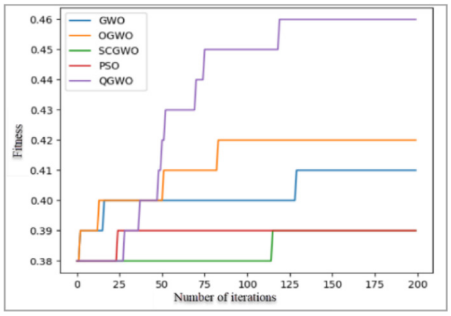

Fig. 10. Comparison of the algorithm fitness value of Group 2.

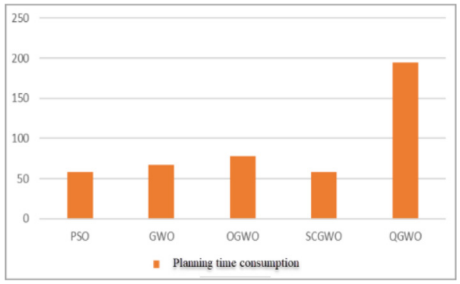

Fig. 11. Comparison of the time consumption of Group 2.

longest time, with a significant difference from other algorithms. Compared with the conventional GWO, it takes two to three times longer. The initial evaluation value of the GWO is lower than that of OGWO, which has the highest fitness in the first generation, reflecting the characteristics of OBL. The line charts for the fitness iteration and the comparison of planning time are shown in Figs. 12 and 13.

(4) Results and analysis of Group 4

As shown in Table 4 Group 4 and Table 10, the QGWO has the highest comprehensive evaluation value, far ahead of the other four algorithms. It can find a suboptimal solution in around 100 generations, while the other algorithms cannot reach the fitness value of the QGWO even in 200 generations. This result can

Table 9
Results of Group 3 algorithm before and after improvement.

Task No.	Planning algorithm	Comprehensive evaluation value	Planning time (s)
Group3	PSO	0.29375	85
	GWO	0.3	140
	OGWO	0.3	134
	SCGWO	0.2875	85
	QGWO	0.30625	275

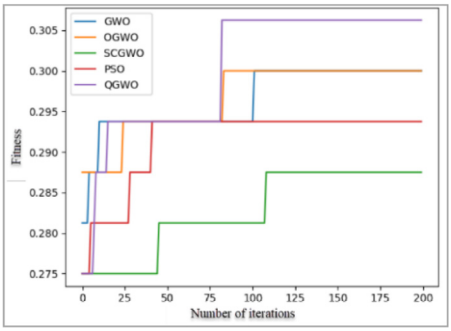

Fig. 12. Comparison of the algorithm fitness value of Group 3.

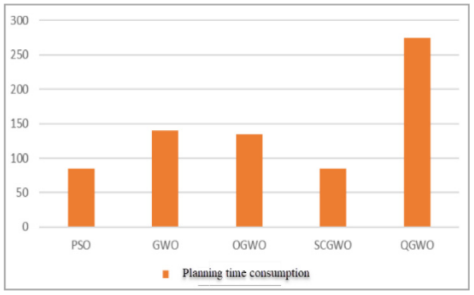

Fig. 13. Comparison of the time consumption of Group 3.

Table 10
Results of Group 4 algorithm before and after improvement.

Task No.	Planning algorithm	Comprehensive evaluation value	Planning time (s)
Group4	PSO	0.245	71
	GWO	0.25	194
	OGWO	0.25	191
	SCGWO	0.25	114
	QGWO	0.275	345

demonstrate the advantages of the QGWO well. In the case of 100 dimensions, it can be found that the other four algorithms have little difference in results after 200 iterations. However, the higher the dimensionality, the more the characteristics of the OGWO can be reflected. The line charts for the fitness iteration and the comparison of planning time are shown in Figs. 14 and 15.

(5) Results and analysis of Group 5

As shown in Table 4 Group 5 and Table 11, the QGWO has the highest evaluation value, and the QGWO takes the most time for planning, with a significant difference from other algorithms. The QGWO performs better in higher dimensions, but after the dimensionality becomes higher, the 200 generations can no longer meet the needs of planning. Meanwhile, the algorithm optimizations has become slow, and the OGWO may not necessarily be able to learn the initial suboptimal solution. This is a shortcoming of the algorithm. The line charts for the fitness iteration and the comparison of planning time are shown in Figs. 16 and 17.

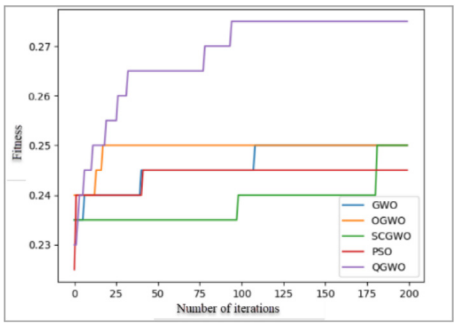

Fig. 14. Comparison of the algorithm fitness value of Group 4.

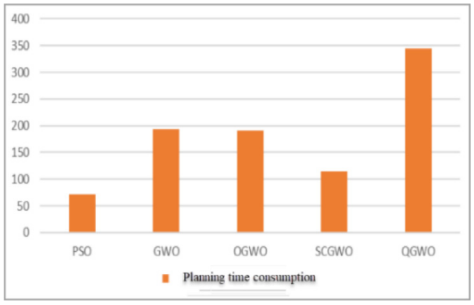

Fig. 15. Comparison of the time consumption of Group 4.

Table 11
Results of Group 5 algorithm before and after improvement.

Task No.	Planning algorithm	Comprehensive evaluation value	Planning time (s)
Group5	PSO	0.2125	117
	GWO	0.2125	242
	OGWO	0.2042	258
	SCGWO	0.2042	165
	QGWO	0.2208	395

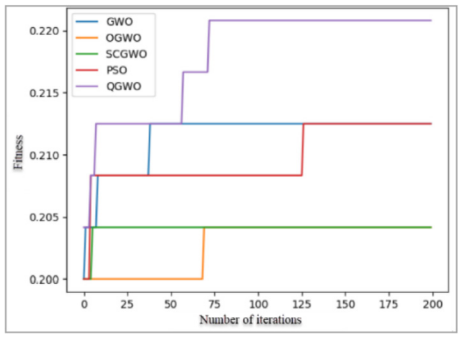

Fig. 16. Comparison of the algorithm fitness value of Group 5.

(6) Results and analysis of Group 6

As shown in Table 4 Group 6 and Table 12, the QGWO has the highest comprehensive evaluation value, and the QGWO takes the longest time for planning, with a significant difference from other algorithms. The performance of the SCGWO is not reflected, but the performance of the GWO is better than of the SCGWO. The line charts for the fitness iteration and the comparison of planning time are shown in Figs. 18 and 19.

Six groups of test data were designed to test the four algorithms mentioned in this paper. The results were compared and analysed to verify the proposed model for tracking resource optimization scheduling. Moreover, the three improved algorithms proposed in this paper were proved to be effective in solving the problem of tracking resource optimization scheduling. In the experiments, the planning results were compared between PSO and the GWO, the OGWO, the SCGWO, and the QGWO. The results indicate that in the model for tracking resource optimization scheduling, compared with the unmodified PSO and the GWO, the QGWO has better performance. Furthermore, the SCGWO can reduce the optimization time, but the performance is poor, and the OGWO and the QGWO deliver the best performance. Both can find suboptimal solutions, but the QGWO takes more time. In the subsequent benchmark function, the algorithm testing was

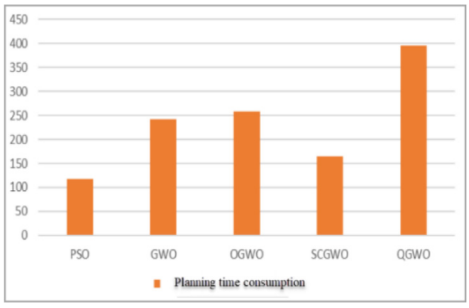

Fig. 17. Comparison of the time consumption of Group 5.

Table 12
Results of Group 6 algorithm before and after improvement.

Task No.	Planning algorithm	Comprehensive evaluation value	Planning time (s)
Group6	PSO	0.173	42
	GWO	0.167	319
	OGWO	0.172	300
	SCGWO	0.173	251
	QGWO	0.18	480

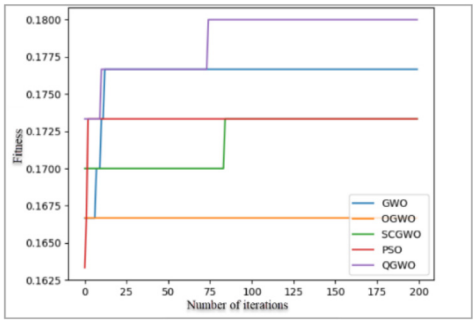

Fig. 18. Comparison of the algorithm fitness value of Group 6.

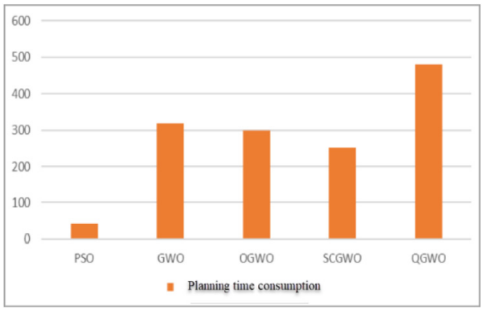

Fig. 19. Comparison of the time consumption of Group 6.

strengthened to reflect the advantages and disadvantages of each algorithm. The results prove the major advantages of the SCGWO in function optimization.

6. Conclusion

At present, the development of optimized scheduling of resources for space debris tracking remains slow. Constraint tracking of targets with limited tracking resources has always been challenging. In this paper, a novel improved GWO was proposed to solve the problem. Three improvement strategies were adapted to the algorithm: using OBL to improve the quality of the initial solution, a sine cosine search strategy to improve the convergence, and reinforcement learning to improve the algorithm's performance. The improved algorithm could converge faster and find the optimal solution through testing compared to state-of-the-art optimization algorithms. The extension of the research will consider resource optimization scheduling for space debris tracking in a complex and dynamically changing environment.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Allworth, J., Windrim, L., Bennett, J., Bryson, M., 2021. A transfer learning approach to space debris classification using observational light curve data. Acta Astronaut. 181, 301–315.

Chaudhari, B., 2022. Role of swarm intelligence algorithms on secured wireless network sensor environment - A comprehensive review. Int. J. Perform. Eng. 18 (2), 92–100.

Cui, Z., Wang, X., 2008. A grid scheduling algorithm based on resources monitoring and load adjusting. In: 2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop Proceedings, KAM 2008. pp. 873–876.

Du, W., Ding, S., 2020. A survey on multi-agent deep reinforcement learning: From the perspective of challenges and applications. Artif. Intell. Rev. 54 (5), 3215–3238.

Emary, E., Zawbaa, H.M., Grosan, C., 2018. Experienced Gray Wolf optimization through reinforcement learning and Neural Networks. IEEE Trans. Neural Netw. Learn. Syst. 29 (3), 681–694.

Fu, M.C., 2019. Simulation-based algorithms for Markov decision processes: Monte Carlo tree search from AlphaGo to AlphaZero. Asia-Pac. J. Oper. Res. 36 (06), 1940009.

Lawal, A.I., Kwon, S., Hammed, O.S., Idris, M.A., 2021. Blast-induced ground vibration prediction in granite quarries: An application of gene expression programming, ANFIS, and sine cosine algorithm optimized ANN. Int. J. Mining Sci. Technol. 31 (2), 265–277.

Liu, J., Tian, M., Huang, P., Lin, Y., Ma, G., 2020. Hybrid decomposition-based algorithm for resource scheduling of remote sensing satellite ground station. J. Jilin Univ. (Sci. Ed.) 58 (3), 611–619.

Mirjalili, S., 2016. SCA: A sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.

Mishra, S., Arora, A., 2022. Double deep q network with huber reward function for cart–pole balancing problem. Int. J. Perform. Eng. 18 (9), 644–653.

Paikray, H.K., Das, P.K., Panda, S., 2021. Optimal multi-robot path planning using particle swarm optimization algorithm improved by sine and cosine algorithms. Arab. J. Sci. Eng. 46 (4), 3357–3381.

Pardeshi, S., Khairnar, C., Alfatmi, K., 2022. Analysis of data handling challenges in edge computing. Int. J. Perform. Eng. 18 (3), 176–187.

Tizhoosh, H.R., 2006. Opposition-based reinforcement learning. J. Adv. Comput. Intell. Intell. Inform. 10 (3), 272–280.

Xu, S., Long, W., 2018. Improved sine-cosine algorithm for solving high-dimensional optimization problems. Appl. Res. Comput. 35 (9), 2574–2577.

Zheng, Y., Yuan, Y., Deng, Y., Li, J., Wang, H., 2016. Satellite resource scheduling algorithm based on pareto front and particle swarm optimization. Comput. Eng. (0): 42 (1), 193–198.

Zhou, B., Wang, A., Fei, C., Yu, W., Zhao, B., 2019. Satellite network resource scheduling mechanism based on reinforcement learning. Comput. Eng. Sci. 41 (12), 2134–2142.

Man Zhao received her B.S. degree, Master degree in Computer Science and Ph.D. in Geoscience information. She is an associate professor in the School of Computer, China's University of Geosciences(Wuhan), Hubei, China. Her current research interests include Intelligent decision-making and planning.

Rui Hou received his M.S. degree in Computer Science in 2021 in the School of Computer, China's University of Geosciences(Wuhan), Hubei, China.

Hui Li received her B.S. degree, Master degree in Computer Science and Ph.D. in Management. She is an professor in China's University of Geosciences(Wuhan), Hubei, China. Her current research interests include Artificial Intelligence and task planning.

Min Ren received his M.S. degree in Computer Science in 2021 in the School of Computer, China's University of Geosciences(Wuhan), Hubei, China.