Multi-robot task assignment algorithm based on improved self-organizing mapping network

GUO Zhenyu\textsuperscript{1,a}, LIU Dan\textsuperscript{2,b}
\textsuperscript{1}Beijing University of Posts and Telecommunications, Beijing, China
\textsuperscript{2}Beijing University of Posts and Telecommunications, Beijing, China
\textsuperscript{a}gzy_work@163.com, \textsuperscript{b}buptld@163.com

Abstract. In order to solve the problem of slow convergence speed and easy task conflict, a multi-robot task assignment algorithm based on improved self-organizing mapping network is proposed in this paper. Considering the redundancy generated in the solution, the self-locking factor is introduced to optimize. The neuron is then reselected when the path intersects to avoid the risk of collision. The simulation results show that the proposed algorithm can effectively reduce the solution time and the distance of multi-robot navigation compared with the traditional algorithm.

1. Introduction
With the construction of China's first space station in 2020, China's space exploration will soon enter a new chapter. At that time, the station will have a number of large equipment, such as solar power stations, space telescopes and giant antennas, which will need to be regularly overhauled to make sure that special parts are functioning properly. Considering its size, manual maintenance outside the crew cabin is extremely inefficient and dangerous. Therefore, it is necessary to use mobile robot for patrol inspection. Robot for patrol inspection can save a lot of time and has a good development prospect\textsuperscript{[1]}. For maintenance tasks of multi-robots, such as reinforcement of the truss support structure of large space antenna, it is necessary to assign all the positions to be reinforced. In other words, according to the characteristics of space missions and under constraints such as time and energy consumption, multiple reinforcement positions are allocated to each robot through algorithms. Each robot is responsible for the maintenance task at multiple locations, completing the truss reinforcement task by successively passing through the reinforcement position and returning to the starting point to complete the task.

At present, multi-robot task allocation methods mainly include behavior-based allocation methods\textsuperscript{[2]}, market auction algorithms\textsuperscript{[3]} and swarm intelligence algorithms\textsuperscript{[4]}. Swarm intelligence algorithms include ant colony algorithm\textsuperscript{[5]}, artificial network algorithm\textsuperscript{[6]}, genetic algorithm\textsuperscript{[7]}, particle swarm optimization algorithm\textsuperscript{[8]}, etc. This kind of algorithm has the advantages of strong adaptability, low error rate, simple implementation, high efficiency, and high solution quality.

In this paper, self-organizing mapping network is used to solve the problem of multi-robot maintenance path allocation under the maintenance problem of large antenna. In order to solve the problem that the traditional multi-task assignment algorithm has a slow convergence speed and is prone to multi-task conflicts, a self-locking mechanism is proposed to optimize the solution process, and new rules are used to reselect neurons when path conflicts occur. Experiments show that the improved
algorithm can effectively accelerate the algorithm convergence, improve the algorithm’s computing efficiency, and effectively avoid path crossing, reducing the risk of robot collision.

2. Model construction

The application scenario of this paper is that according to the result of pre-planning, the multi-autonomous on-orbit service robot is utilized to inspect the position in turn, perform close maintenance and reinforcement service operation, and then return. During the whole process, the robots do not collide with each other and try to avoid waiting for a long time to complete the assembly task with high efficiency.

2.1 Problem model construction

Firstly, a set $C = \{c_i | i = 1, 2, ..., m\}$ consisting of $m$ positions to be serviced and a set $R = \{r_j | j = 1, 2, ..., n\}$ consisting of $n$ robots are established. Each robot completes all tasks and returns to the starting point with a voyage cost $L = \{l_j | j = 1, 2, ..., n\}$.

After algorithm allocation, $r_j$ will be assigned to a maintenance execution sequence set $\varphi_j$, then the target positions of $r_j$ execution tasks are:

$$\varphi_j = \left\{ \left( x_{0j}, y_{0j} \right), \ldots, \left( x_{ij}, y_{ij} \right), \ldots, \left( x_{nj}, y_{nj} \right) \right\}$$

(1)

Where, $\left( x_{0j}, y_{0j} \right)$ is the starting position of $r_j$, and $\left( x_{ij}, y_{ij} \right)$ is some intermediate overhaul position of $r_j$.

Task assignments need to prioritize the following constraints:

1) Reduce the travel distance of the robot during its mission.
2) Reduce the planning time used to allocate tasks.
3) For safety reasons, the distribution results should minimize the cross paths.

2.2 Performance evaluation index

The multi-robot task assignment scheme can be evaluated by multiple indicators, and each performance indicator will cross and cover, so there is no unique global optimal solution. This paper mainly optimizes the algorithm efficiency, so the mathematical model established in this paper will consider the convergence index of the algorithm (including the convergence speed of neurons and CPU computing time) and the voyage cost required to complete the task, so as to obtain the performance index function of on-orbit intelligent assembly task assignment:

$$\min H = \sum_{j=1}^{n} L_j + \varepsilon T = \sum_{i=1}^{n} \sum_{a=1}^{q} l_{j,a} + \varepsilon T$$

(2)

Where, $l_{j,a}$ represents the voyage distance cost corresponding to the robot $r_j$ performing the $c_{ja}$ task. $T$ is the CPU time consumed by the algorithm. $\varepsilon \in (0,1)$ is the weight of specific gravity, which is used to normalize $T$.

3. Algorithm

3.1 Traditional SOM algorithm

The core of SOM algorithm is competitor mechanism and weight update rule, and the expression of competitor mechanism is:

$$v^* = \min |D_q| \times \left( 1 + \frac{l_{q} - l_{avg}}{l_{avg}} \right)$$

(3)
Where, \( D_{ij} \) is the Euclidean distance between the \( ith \) overhaul position of the input network and the \( jth \) neuron of the output layer. \( l_j \) is the length of the winner's ring, and \( l_{avg} \) is the average length of all rings. The rules for neuron renewal are as follows:

\[
V'_i = V_j + \mu \times f(d, G)(c_i - V_j)
\]  

(4)

Where, \( \mu \) is the learning rate of the algorithm, and the network gradually converges through independent learning. \( f(d, G) \) is the neighborhood function, and the expression is:

\[
f(d, G) = \begin{cases} 
  e^{-d^2/G^2} 
  & \text{if } d < r \\
  0 
  & \text{otherwise}
\end{cases}
\]

(5)

Where, \( G \) is the gain parameter, \( d \) is the distance gain parameter between each node and the winner, and \( G \) is the decreasing function of time, which is expressed as:

\[ G = (1 - \alpha)G_0 \]

(6)

Where, \( \alpha \) is the update rate, ensuring that the algorithm has a certain convergence rate. \( G_0 \) is the initial parameter of the neighborhood.

3.2 SOM algorithm improvements

In the case of multi-task assignment, the traditional SOM algorithm is easy to carry out redundant calculation, resulting in slow solution speed. When the task volume is large, there will be task conflicts, resulting in no solution planning. The improvement of this problem in this paper is as follows:

1) In the task input phase, a locking factor is introduced to determine whether to lock the task and the winner according to whether the task is the ideal input. After locking, multiple tasks can avoid attracting neurons at the same time and thus causing interference. The rule is: each output neuron can only choose whether to lock the task or not. After locking, the algorithm removes task from the alternative list, defines the travel time used for movement as the locking cost and adds the task execution cost.

After improvement, the neighborhood function \( f(d, G) \) is modified as follows:

\[
f(d, G) = \begin{cases} 
  e^{-d^2/G^2} \times k 
  & \text{if } d < r \\
  0 
  & \text{otherwise}
\end{cases}
\]

(7)

Where, \( k \) is the locking factor, when \( r_j \) locks target task \( c_i \), the value of \( k \) is set as 1; otherwise, it is set as 0.

After adding the locking cost, the target function is modified as:

\[
C = \sum_{j=1}^{n} \left( c_j + \sum_{k=1}^{k} m_{jk} \right)
\]

(8)

Where, \( k \) is the current task index of \( r_j \), \( g \) is the number of assigned maintenance tasks of \( r_j \), and \( m_{jk} \) represents the locking cost, which is calculated by the following formula:

\[
m_{jk} = \begin{cases} 
  \text{dis}_{k,j} - \min(D), & \text{locked} \\
  +\infty, & \text{unlocked}
\end{cases}
\]

(9)

Where, \( \text{dis}_{k,j} \) represents the distance between maintenance position \( k \) and robot \( r_j \), and \( \min(D) \) represents the minimum distance between the remaining position and robot \( r_j \).

2) As the traveling route of the robot, there will be the danger of collision between robots when the routes intersect. In order to avoid crossing paths, some improvements are made to the algorithm. As shown in Fig 1, when assigning tasks, the distance priority principle is usually followed, so that the selected neuron will cross the allocated path when moving to the task in some cases. Meanwhile, the
winner drives the neighboring neuron to move, which will cause a lot of distance recalculation, thus reducing the computational efficiency of the algorithm.

Two sets $C^p_j$, $C^o_j$ were established. $C^p_j$ represents the set of distance from the current overhaul position $c_j$ to the nearby neurons. When a path intersects, the length of the path being intersected constitutes $C^o_j$. When $R_i$ is selected, $R_i$, $R_b$ is moved along with it, causing the related cost to be recalculated. At this time, the improved algorithm reselects $R_2$ which is closer to the target point on the cross path.

3.3 Algorithm process

After improvement, the whole algorithm flow is as follows:

Step 1: Initialize the initial position $W$, network update rate $\alpha$, network learning rate $\mu$ and locking cost $m$ of the robot;

Step 2: Randomly select a maintenance position $T$ to input into the network;

Step 3: Obtain the winning neuron and its neighborhood according to the competitor mechanism and neighborhood rules;

Step 4: Determine whether the locking condition is met and conduct the locking;

Step 5: Judge whether there is path crossing, then proceed to step 6; otherwise, skip step 6 and proceed to step 7;

Step 6: Select the neuron moving closer to the target on the cross path;

Step 7: Update the weight of the moving neuron and the neighboring neuron;

Step 8: Repeat steps 2 through 7 until all repair locations are assigned and the algorithm is complete.

4. Simulation and analysis

In order to verify the performance of the improved algorithm, the simulation experiment of the algorithm was conducted on the Windows10 operating system based on MATLAB 2018b environment. The PC is configured with an Intel(R) Core i7-7700@3.60ghz processor and 8G of ram.

The experimental parameters are set as follows. In the working area of the robot, 80 target positions that need to be repaired are set. The three robots start from the same starting point, go through the assigned target maintenance successively and finally return to the starting position. The relevant parameters of the algorithm are set as: $\alpha = 0.03$, $\mu = 0.6$, $m = +\infty$. The SOM algorithm and the improved algorithm were used for task assignment respectively. The task points and routes assigned by each robot are shown in Fig 2 and Table 1.

Fig 2(a) is the allocation result of SOM algorithm, and (b) is the result of the improved algorithm. It can be seen that the three robots start from the starting point and return to the starting point after completing the task at all the task points. However, there are two intersecting paths in the distribution route of SOM algorithm, and no intersecting paths are generated in the improved algorithm, indicating that the improved algorithm has optimized and reselected the intersecting paths. It can be seen from table 1 that the total distance needed by the robot solved by the improved algorithm to complete the task...
is less than the result of SOM algorithm, and the path length is shortened by 8.9%, indicating that the improved algorithm can better reduce the transportation cost of the robot.

In order to verify the improved algorithm's improvement on the algorithm's computing efficiency, 4 groups of 50 task assignment experiments were carried out with the two algorithms respectively, and the average allocated completion time was calculated respectively. The results are shown in Fig 3. The number of neurons in the task assignment experiment statistical algorithm of 200 was taken, as shown in Fig 4.

As can be seen from Fig 3, in the 4 groups of experiments, the CPU computing time of the improved algorithm was all less than that of SOM algorithm, indicating that the efficiency of the improved algorithm for task assignment solution was improved. As can be seen from Fig 4, the neuron of the improved algorithm converges in about 310 seconds, while the original SOM neuron converges in about 420 seconds, with the convergence rate increased by 26%, indicating that the improved algorithm converges faster.

5. Conclusion
In this paper, a multi-robot task assignment algorithm for multi-point maintenance of truss is proposed. To solve the problem of slow convergence of traditional self-organizing mapping network algorithm, self-locking factor is introduced to optimize the solution process and reduce redundant calculation. The new rules are then used to reselect the robot when the paths intersect so as to avoid path conflicts and reduce unnecessary cost calculations. Simulation results show that compared with the original SOM algorithm, the improved algorithm can effectively improve the problem of slow convergence speed, avoid path crossing and improve the quality of task assignment.
Reference

[1] DAI Z D. Progress and Key Technologies in Several Frontiers of Space Robots [J]. MANNED SPACEFLIGHT, 2016,22(01):9-15.
[2] LI G J. Task allocation of warehouse robots based on intelligence optimization algorithm[D]. Harbin: Harbin Institute of Technology, 2013.
[3] LIU L, JI X C, ZHENG Z Q. Multi-robot task allocation based on market and capability classification[J]. Robot, 2006, 28(3): 337-343.
[4] LI X M, YAN J, LIU B. A Survey of Multi-Agents Cooperative Task Allocation Research [J]. Computer and Digital Engineering, 2014(12):227-234.
[5] Dorigo M, Blum C. Ant colony optimization theory: A survey[J]. Theoretical Computer Science, 2005, 344(2-3):243-278.
[6] Kohonen T, Schroeder M R, Huang T S. Self-Organizing Maps[M]. Springer Berlin Heidelberg, 2001.
[7] Holland J H. Genetic algorithms[J]. Scientific American, 1992, 267(1):66-72.
[8] Kennedy J, Eberhart R. Particle swarm optimization[C] Proceedings of ICNN'95 - International Conference on Neural Networks. IEEE, 1995.