energies

MDPI

Article

An Improved SOM-Based Method for Multi-Robot Task Assignment and Cooperative Search in Unknown Dynamic Environments

Hongwei Tang [1,2,*], Anping Lin [3,*], Wei Sun [2] and Shuqi Shi [1]

[1] Hunan Provincial Key Laboratory of Grids Operation and Control on Multi-Power Sources Area, Shaoyang University, Shaoyang 422000, China
[2] College of Electrical and Information Engineering, Hunan University, Changsha 410082, China
[3] College of Electronic Information and Electrical Engineering, Xiangnan University, Chenzhou 423000, China
* Correspondence: tanghongwei@hnu.edu.cn (H.T.); Anping719@126.com (A.L.)

Received: 24 April 2020; Accepted: 25 June 2020; Published: 26 June 2020

Abstract: The methods of task assignment and path planning have been reported by many researchers, but they are mainly focused on environments with prior information. In unknown dynamic environments, in which the real-time acquisition of the location information of obstacles is required, an integrated multi-robot dynamic task assignment and cooperative search method is proposed by combining an improved self-organizing map (SOM) neural network and the adaptive dynamic window approach (DWA). To avoid the robot oscillation and hovering issue that occurs with the SOM-based algorithm, an SOM neural network with a locking mechanism is developed to better realize task assignment. Then, in order to solve the obstacle avoidance problem and the speed jump problem, the weights of the winner of the SOM are updated by using an adaptive DWA. In addition, the proposed method can search dynamic multi-target in unknown dynamic environment, it can reassign tasks and re-plan searching paths in real time when the location of the targets and obstacle changes. The simulation results and comparative testing demonstrate the effectiveness and efficiency of the proposed method.

Keywords: multi-robot; self-organizing map; dynamic window approach; task assignment and cooperative search; unknown dynamic environments

1. Introduction

Multi-robot systems have become a prominent research area and have attracted increasing attention from researchers. A multi-robot system can be used in many applications, such as in warehousing, search and rescue missions, ground-cleaning, industrial transportation, etc. Task assignment and path planning of a multi-robot system are fundamental issues in the research field of multiple robots. An integrated biologically inspired self-organizing map (BISOM) algorithm was proposed for a 3-D underwater environment [1]. In unknown environments with obstacles, how to assign a group of robots to a target's location and achieve autonomous obstacle avoidance remains a challenging topic in multi-robot systems [2–4].

The task assignment of multi-robot systems has attracted the attention of many researchers. Wei et al. [5] studied the multi-robot task allocation problem in the search and retrieval domain. Liu et al. [6] introduced a task allocation mechanism that is based on the ant colony optimization of a swarm intelligence algorithm, which can effectively achieve loosely coupled and tightly coupled task allocation in large-scale multi-robot systems. Yao et al. [7,8] studied task allocation and path planning, such as offline route planning for a coverage search mission in a river region and the time-optimal trajectory generation for aerial coverage of urban buildings. The self-organizing map

(SOM) is an unsupervised neural network for data visualization that was first proposed by Kohonen [9]. Some SOM-based methods have been used for energy management, such as battery sorting [10], evaluation of energy distribution [11], efficient utilization of resources [12], etc. Some SOM-based approaches have been proposed to resolve the task assignment for multi-robot systems [13–15]. The task assignment of a multi-robot system is acquired based on the competition principle of the SOM, and the path planning of each robot is realized by adjusting the weight updating rules of the SOM. However, these studies have focused on establishing a control model based on prior environmental information at the lowest cost, and seldom consider the path planning problem encountered by each robot en route to its destination in real environments.

The purpose of task assignment and path planning is to arrange the robots so that they visit all of the assigned target locations, while ensuring that the robots can avoid obstacles smoothly. Path planning is required after task assignment, and some improved algorithms have been proposed. However, as the number of robots increases, so too does the computational complexity of the algorithm, such as the artificial bee colony [16], the A* algorithm [17], and the genetic algorithm [18]. A multi-robot algorithm was proposed for exploration, which is better than a depth-first search with a single robot [19]. The artificial potential field (APF) [20] is an efficient path planning algorithm [21,22]. This method has the advantage of low computational complexity, and it has long been proposed in the field of robot path planning. Closed-loop control of the current path and environmental information are utilized to smooth the trajectory. However, the well-known drawbacks of the APF, such as a low efficiency and the local minimal problem, have prevented its widespread application, and the APF is difficult to use in a dynamic environment with unknown location information of obstacles.

Most methods of task assignment and path planning are mainly focused on environments with prior information. Since most algorithms use a global search approach, these methods are suitable for static environments, not unknown dynamic environments with obstacles. Local path planning is an important part of the autonomous motion of mobile robots and requires that a robot moves towards the preferred location without collisions in a completely or partially unknown environment. The dynamic window approach (DWA) is an efficient real-time planning algorithm that was first proposed by Fox D et al. [23]. The DWA has been successfully applied in robot operating system (ROS) navigation; the robot's motion trajectory can be adjusted according to the environmental information acquired in real time. In recent years, many obstacle avoidance methods have been proposed; compared with other obstacle avoidance methods, the DWA accounts for the dynamic properties and the constraints of real robots, such as the maximum velocity, acceleration, etc. Therefore, the control instructions obtained by this method appear to be more in line with the actual situation [24].

In this paper, in unknown dynamic environments where the location information of obstacles requires real-time acquisition, an improved DWA is introduced into an improved SOM neural network to solve the problem of task assignment and cooperative search for multi-robot systems, the proposed method is called the DSOM method. The proposed DSOM method has three main contributions. First, to avoid the robot oscillations and hovering that appear when using an SOM-based algorithm, an improved SOM neural network with a locking mechanism is developed to assign the robots in order to access multiple targets. Second, an adaptive DWA is introduced to adjust the velocities of the robots in order to update the weights of the winning neurons and their neighboring neurons in the SOM in such a way that the robot can automatically avoid speed jumps and obstacles, and achieve a better planning search path in unknown environments. Third, the proposed method can reassign tasks and re-plan the search paths in real time when the location of the target and obstacle changes. The robot acquires real-time distance information with obstacles through sensors, and then it adjusts the linear and angular velocities of the robot through the adaptive DWA to achieve path planning in unknown dynamic environments.

This paper is organized as follows. The problem statement is briefly introduced in Section 2. In Section 3, the traditional approaches are introduced. Section 4 describes the proposed method of the DSOM. In Section 5, the experiments are described. A brief conclusion is provided in Section 6.

2. Problem Statement

In this study, the environmental information is initially unknown. We assume that each robot is equipped with a wireless network device and sensors to measure the relative bearing. The proximity information comes from the local sensors and none of the sensors are faulty. The robot is searching in the unknown environment, the robot can only sense the nearby robots and obstacles through sensors, and it shares this information across the network. We use this information to construct and dynamically update the network topology [25].

The essential issue of task assignment and cooperative search in an unknown dynamic environment is to decompose all the tasks into multiple subtasks according to the perceived environmental information so that all the robots can find their optimal path to move to the specified target and achieve obstacle avoidance. A robot can only perform one task at a time, and each target requires at least one robot to visit, but it can continue to perform another task after completing the first task. There are three primary requirements for the task assignment and cooperative search of multi-robot systems in unknown dynamic environments: (1) each task is assigned to an appropriate robot at the cost of minimizing a multi-robot system; (2) all robots can find the optimal path safely and collision-free when completing the assignment task, and in the overall motion of the robot, collisions with obstacles or other robots are not allowed; and (3) robots are equipped with sensors, and the mobile robots have basic capabilities for communication and location recognition.

Although there have been some SOM-based task assignment and cooperative search studies in multi-robot systems, there are still some issues that need to be improved: (1) the traditional SOM-based method has a phenomenon of output neuron oscillation. In some cases, this phenomenon will lead to too many iterations of the algorithm and even cause the system to fall into infinite iterations, resulting in poor robustness of the system and unsmooth path planning. (2) In some unpredictable emergencies, such as when some robots failed, the robustness of traditional algorithms is poor. Additionally, robots often encounter the sharp speed jump problem; (3) In an environment where the locations of the obstacles are unknown or in a dynamic unknown environment, the effect of obstacle avoidance in the robot cooperative search is not ideal.

3. Traditional SOM and Motion Model of DWA

3.1. SOM Structure of Multi-Robot System

The SOM is generally composed of an input layer and an output layer, which are connected by weight between the output layer and input layer. The SOM structure for task assignment and path planning of a multi-robot system is shown in Figure 1. This model only has two layers of neurons; namely, the input layer (targets' locations) and the output layer (robots' positions). The input layer consists of two neurons, where the two-dimensional input vector (x_i, y_i) represents the Cartesian coordinates of the ith task. The output layer (robots' positions) is a competitive layer, in which each neuron represents the location of a robot and is connected to the neurons in the input layer (targets' locations). The connection weight between the neurons in the output layer and the input layer is given by a two-dimensional weight vector. The SOM includes the winner selection rules, the definition of the neighborhood function and the weights update rules. The SOM-based method is used to select winners for the target locations. In addition, the neighborhood function is used to select the neighbor of the winning robot, and the moving speed of the winner and its neighbor is determined. Finally, by updating the weight vectors of the winner robots, the robots can reach the target locations along collision-free paths. When all of the target positions have been accessed, the task assignment and path planning are completed [1].

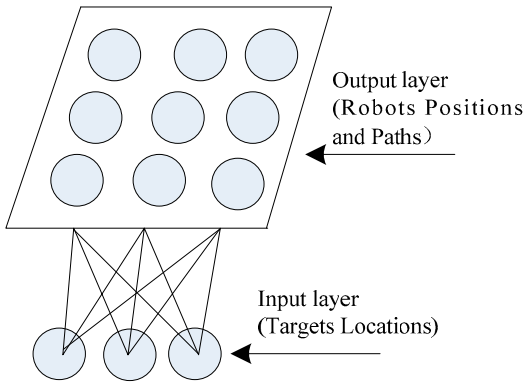

Figure 1. Self-organizing map (SOM) structure of a multi-robot system.

3.2. Traditional SOM for Task Assignment and Path Planning

3.2.1. Winner Selection Rules

For a given input neuron (target coordinate) $T_i = (x_i, y_i)$, the competitive layer neurons compete to be the winner. According to the winner selection rules, the robot with the shortest Euclidean distance from the input task is selected. The rules are defined as follows:

$$D_{ij} = \left| T_i - R_j \right| = \sqrt{(x_i - w_{jx})^2 + (y_i - w_{jy})^2} \tag{1}$$

$$W \Leftarrow \min\left\{ D_{ij} \right\} \quad for \quad W \in \Omega \tag{2}$$

where D_{ij} represents the Euclidean distance between the ith input neuron (task) and the jth competitive neuron (robot). $R_i = (w_{jx}, w_{jy})$ represents the coordinates of the jth competitive neuron. W is the selected winner neuron to the ith input neurons. To prevent the robot from being the winner more than once in each iteration, Ω is a set of nodes that provides all of the possible choices for $T_i = (x_i, y_i)$. In traditional SOM task assignment for 3D coordinates, the difference is that the winner selection rules are based on 3D coordinates, while other cases are consistent with the application of 2D environment [1,14,15].

3.2.2. Definition of the Neighborhood Function

After the winner selection is finished, the next step is to find the neighbors of the winner. A neighborhood function is introduced to define the influence of the input data on the winner and its neighboring neurons. The neighborhood range is defined as a circle with radius r, where the center is the location of the winner robot. The neighborhood function determines the attractive strength of the input target on the winner as well as its neighbors. The attraction force on the selected neurons is diminishing as the distance between the neighbors and the winner increases. The neighborhood function is described as:

$$f(d_j, G) = \begin{cases} e^{-d_j^2 / G^2(t)}, & d_j < r \\ 0, & others \end{cases} \tag{3}$$

where $d_j = \left| R_n - R_j \right|$ is the distance between the nth output neuron and the winner. The nonlinear function $G(t)$ is calculated by $G(t) = (1 - \alpha)^t G_0$, where t is the number of iterations, G_0 is the gain parameter, α is the update rate determining the calculation time and $\alpha < 1$ is constant. The calculation time decreases with an increase in the parameter α.

3.2.3. Weights Updating Rule

The winners and their corresponding neighbors will move along effective paths. A Self-organizing neural network not only allocates tasks to suitable robots but also plan paths for each robot to the assigned tasks by updating the weights. The update rule is described by the following equation:

$$R_j(\mathrm{n}+1) = \begin{cases} T_i, & D_{ij} < D_{\min} \\ R_j(n) + \beta \times f(d_j, G) \times (T_i(n) - R_j(n)), & others \end{cases} \quad (4)$$

where $D_{\min}$ is the termination condition of iterations ($D_{\min}$ can effectively decrease the calculation time). As long as $D_{ij} < D_{\min}$, the weight value of the jth competitive neuron will be replaced by the ith input vector, which means that the jth robot has reached the ith task. The learning speed β has an effect on the modification of the weight and the computing time [13].

3.3. Motion Model of DWA

The DWA is an obstacle-avoidance approach that accounts for the dynamic and kinematic constraints of a mobile robot. The concept of the DWA is to maximize the objective function and directly obtain an optimal velocity command in the velocity space. The position control of the robot is transformed into speed control, and the obstacle avoidance problem is described as an optimization problem with speed space constraints [23,24]. There are two types of constraints: (1) the velocity and acceleration constraints of the robots; and (2) the environment and obstacle constraints.

The velocity command is obtained from the velocity space of the robot. Three kinds of velocity spaces are defined in the DWA, which can be expressed as V_s,V_a and V_d. Each velocity in the velocity space is a tuple (v, w), in which v is the translational velocity and w is the rotational velocity. V_s is composed of all of the possible velocities the robot can achieve.

$$V_s = \{(v, w) \big| v \in [v_{min}, v_{max}]; w \in [w_{min}, w_{max}]\} \quad (5)$$

Each velocity in V_a ensures that if we choose this velocity, the robot can stop before crashing into any obstacle. Here, $\dot{v}_b$ and $\dot{w}_b$ are the accelerations for breakage.

$$V_a = \left\{ (v, w) \middle| \begin{array}{l} v \le \sqrt{2.dist(v, w).\dot{v}_b}; \\ w \le \sqrt{2.dist(v, w).\dot{w}_b}. \end{array} \right\} \quad (6)$$

V_d is called the dynamic window, which contains the velocities that can be reached in a clock cycle. Here, v_c and w_c are the actual velocities. $\dot{v}_a$ and $\dot{w}_a$ are the maximum acceleration, and $\dot{v}_b$ and $\dot{w}_b$ are the maximum decelerations.

$$V_d = \left\{ (v, w) \middle| \begin{array}{l} v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t] \wedge \\ w \in [w_c - \dot{w}_b \Delta t, w_c + \dot{w}_a \Delta t] \end{array} \right\} \quad (7)$$

Then, the resultant velocity search space is as follows:

$$V_r = V_s \cap V_a \cap V_d \quad (8)$$

Each velocity in V_r will be sampled and its appropriateness is calculated with the objective function. The objective function is defined as follows:

$$G(v, w) = \sigma(\alpha.heading(v, w) + \beta.dist(v, w) + \gamma.vel(v, w)) \quad (9)$$

where α, β, and γ are constant, and usually α+β+γ=1. Here, σ is a smoothing constant.

In these functions, $heading(v, w)$ is an azimuth evaluation function, which maintains the course of the robot towards its target location and is maximal if the robot moves directly towards the target; $dist(v, w)$ is the distance to the closet obstacle on the trajectory, for which the nearer the obstacle is, the more the robot will be inclined to move around it; and $vel(v, w)$ is the linear velocity to force the robot to move fast. A robot's velocity is a tuple (v, w), and includes the linear velocity and angular velocity—that is, the magnitude and the direction of the velocity. When we simulate the trajectory of the robot, the robot finds the positions of the obstacle and other robots, calculates the distance between the robot and the obstacle or other robot, and then judges whether the current sampling velocity can stop before hitting anything else. If it can stop, it is admissible; otherwise, it will be abandoned.

4. Proposed Method of DSOM

4.1. Adaptive DWA

The proposed algorithm in this paper assumes that the robot is equipped with a ranging sensor, which can measure the distance and bearing from the obstacle within a certain range. The safe distance between the robot and the obstacle or other robots is set as D_d. The dynamic weight of the linear velocity is defined as the adaptive dynamic weight γ_v.

$$\gamma_v = \begin{cases} \gamma_{\min} + \psi(\frac{dist(v,w)}{D_{\min}})\kappa, & dist(v, w) < D_d \\ \gamma_{\max} \quad , & dist(v, w) > D_d \end{cases} \tag{10}$$

where $\gamma_{\min}$ is the minimum weight and $\gamma_{\max}$ is the maximum weight. The weight range for the adaptive dynamic weight is set to [$\gamma_{\min}$,$\gamma_{\max}$], ψ and κ are constants, and κ = 1.2 in this paper.

The motion model of DWA is still used in the adaptive DWA, we only improve Equation (9), and define the objective function as follows:

$$G(v, w) = \sigma(\alpha.heading(v, w) + \beta.dist(v, w) + \gamma_v.vel(v, w)) \tag{11}$$

In this paper, $\alpha = 0.6$, $\beta = 0.2$ and $\sigma = 1$. To avoid the high proportion of one of the values, all three components of the objective function are normalized to [0,1]. By combining all three of these functions, the robot avoids the collision as quickly as possible under the abovementioned constraints while still moving in the direction of reaching the target.

Finally, the velocity command for the next time period is the highest velocity of the objective function value. In other words, we have

$$V_t = \text{argmax}\{G(v, w)\} \tag{12}$$

4.2. DSOM for Task Assignment and Cooperative Search

4.2.1. The Proposed Locking Mechanism

In the process of multi-robot task allocation, the output neuron oscillation phenomenon exists in some SOM-based neural network method. The definition of the output neuron oscillation phenomenon in our study is as follows: Output neuron oscillation refers to when the same output neuron is attracted by different targets in such a way that the output neuron moves back and forth between those targets [26]. When there are more targets than robots, the probability of this phenomenon occurring is relatively high, and the worst result occurs when the output neuron always moves back and forth between the input targets and cannot achieve any target. Such a result could lead to too many iterations and even failure of the task allocation. Because of the randomness of the input targets, there is a possibility that the output neuron (robot) could never visit any target.

To illustrate the working process of selecting of the winner and neighbors as well as the locking mechanism, an example is shown in Figure 2, where the blue dots represent the initial locations of the

robots, the black circles represent the obstacles, and the hexagonal stars denote the positions of the targets. In this study, we assume that there is no difference between each pair of robots, which means that they all have the same parameters and functions. When all of the targets in the workspace are visited by the robots, all of the tasks are completed. For each robot, the cost required to complete all the assigned tasks for the robot is defined as the time cost (number of iterations) and the path length. The total cost is the sum of the path length of all the robots and the maximum iteration times among all of the robots.

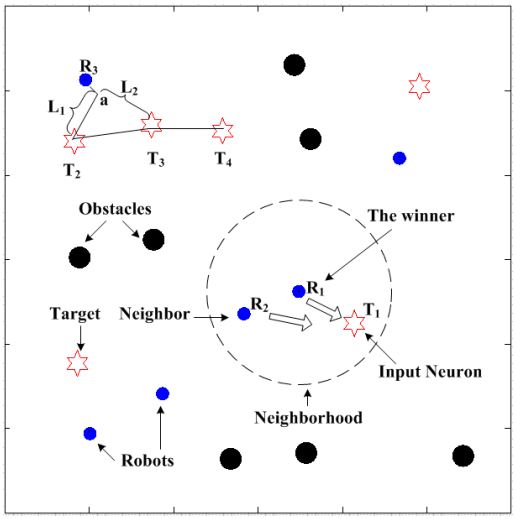

Figure 2. The process of the SOM with the locking mechanism.

In Figure 2, it is assumed that the location of target T_1 is one of the input vectors, and robot R_1 becomes the winner according to the winner selection rules. Then, we define the winner's neighborhood domain (with radius r). Obviously, robot R_2 is in the neighborhood area. Finally, according to the weight updating rule in Equation (4), robots R_1 and R_2 will move towards target T_1 at different speeds. Additionally, there are three targets and one robot in the top left corner. Regardless of whether target T_2, target T_3 or target T_4 are used as the input data for the network, the winner is robot R_3, and there are no neighbors in the neighborhood of the winner. By analyzing the character of the winner selection rules in Equation (2) and neighborhood updating rule in Equation (3), the output neuron (robot) will move back and forth between targets T_2, T_3 and T_4.

To avoid the oscillation and hovering phenomenon of the output neuron (robot), a locking mechanism is proposed in this paper to improve the SOM-based algorithm. The mechanism is described as follows: each output neuron (robot) can either lock a target or not lock a target, and only one target can be locked by an output neuron (robot). Once an output neuron (robot) locks a target, the output neuron will not move towards other targets. The time from the current position of the robot to the target location is called the locking cost of output neuron R_i and target T_i. Two requirements are needed to lock the output neuron R_i and the target T_i. First, when T_i is the input data for the network, R_i is the winner neuron and there are no other neurons in the neighborhood. Second, the locking cost must be minimal. Take Figure 2 as an example to further illustrate the principle of locking mechanism. After neural network initialization, robot R_3 does not lock any target, thus the locking cost of R_3 is infinite. When T_3 is randomly selected as the input neuron, according to the winner selection rules, R_3 is called the winner neuron and there is no other robot in the neighborhood, and the distance from R_3 to T_3 is less than the infinity in the locking table, which means that the two requirements of locking mechanism are met, and the locking target in the locking table is T_3. After an algorithm iteration, when R_3 moves to point a, the locking cost of R_3 is updated to L_2. In the second iteration period, T_2 is randomly selected as the input neuron, and only R_3 is in the neighborhood. We assume that the distance from point a to T_2 is less than that to T_3, then the two requirements are met, and then the

locking target of R_3 is modified to T_2, the locking cost is updated to L_1. In the next iteration period, whether T_3 or T_4 is randomly selected as the input neuron, the locking cost of robot R_3 to T_3 or T_4 is more than that of robot T_2, so the locking target of R_3 remains unchanged, until R_3 reaches T_2, the locking target of R_3 is reset to null. After that, repeating the above process, T_3 and T_4 will be locked successively.

If a robot has locked a target, it is only required to satisfy requirement 1. When robot R_j locks one target rather than the current input target T_i, the value of $L(j)$ is reset to 0. Otherwise, the value of $L(j)$ is reset to 1.

$$L(j) = \begin{cases} 0, & T \text{ is not } T_i \\ 1, & others \end{cases} \tag{13}$$

4.2.2. New Neighborhood Function

After selecting the winning robot, the next step is to find the winner's neighbors. The influence of the input data on the winner and its neighboring neurons is defined by a neighborhood function. Taking the position of the winner robot as the center, the neighborhood range is defined as a circle with radius r. The attractive strength of the input target on the winner robot as well as its neighbors is determined by the neighborhood function. The attraction force on the selected neurons diminishes as the distance between the neighbors and the winner increases. Because of the locking mechanism, the neighborhood function in Equation (3) is redefined as follows:

$$f(d_j, G) = \begin{cases} e^{-d_j^2/G^2(t)} \times L(j), & d_j < r \\ 0, & others \end{cases} \tag{14}$$

According to the locking mechanism, $L(j)$ is a function in Equation (13) with a value range of $\{0, 1\}$.

4.2.3. New Weights Updating Rule with DWA

The tasks are assigned to the appropriate robots via the SOM neural network, and the paths for the assigned tasks are planned by updating the weights. The winners of the robots and their neighbors will move forward the along optimized paths. To enable the robot to achieve autonomous obstacle avoidance in the process of accomplishing the task, this paper redefines the neighborhood updating rule of the SOM based on the motion model in the DWA. The weight updating rule in Equation (4) is redefined as follows:

$$R_j(n+1) = \begin{cases} T_i & D_{ij} < D_{\min} \\ R_j(n) + \rho \times f(d_j, G) \times |V_t| & others \end{cases} \tag{15}$$

where $|V_t|$ is required to be suitable data in order to satisfy the speed control constraints of the robot, and therefore, the robot does not exhibit a sharp speed jump phenomenon. The learning speed ρ has an effect on the computing time and the number of iterations; the larger the value of ρ is, the faster the robot is.

In the new weights updating rules, the motion model in the DWA is applied to the definition of the new rules. Robots can adjust the linear and angular velocities in real time through the DWA according to the information acquired by sensors in the environment, which thus improves the response ability to the environmental changes and avoids the speed jump problem, as well as improving the cooperative search in the environments where the obstacle locations are unknown. The DSOM for task assignment and cooperative search can be implemented to avoid static or dynamic obstacles.

4.3. Implementation of the DSOM

The flow diagram of the DSOM method is shown in Figure 3. Next is a detailed description of the proposed method of the DSOM. The DSOM method is obtained following the basic steps below.

Step 1: Initialization. Initialize the weights of the output neuron (robots' position), input neuron (tasks' position), and locking table;
Step 2: A task T_i is randomly selected and input into the neural network;
Step 3: According to winner selection rules and the neighborhood rules, the winner neurons (winner robots) and their neighbors are found;
Step 4: Update the locking table according to the locking mechanism;
Step 5: Compute the velocity $|V_t|$ and update the robots' position of the winner robots and their neighbors by the adaptive DWA;
Step 6: Update the locking cost;
Step 7: Repeat steps 2 to 6 until all tasks and the program are completed.

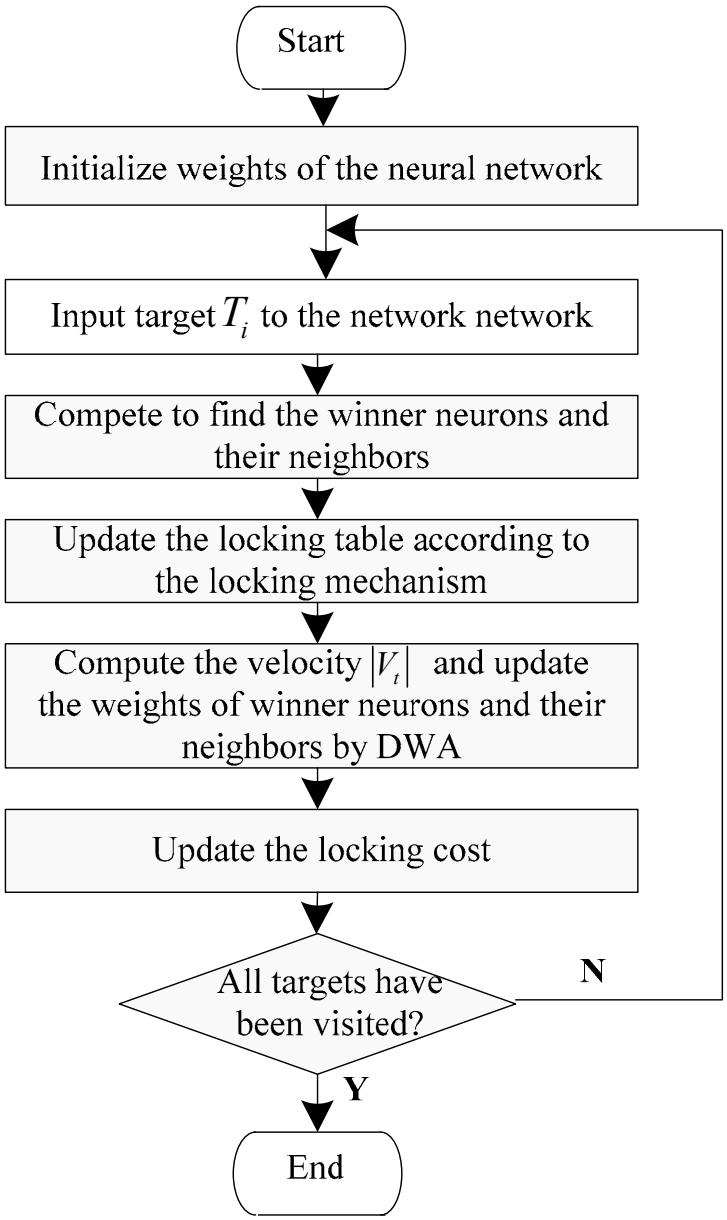

Figure 3. Flow diagram of the DSOM.

5. Validation and Comparison

In this paper, the effectiveness and efficiency of our proposed DSOM method was verified by a series of simulation experiments. All simulations were run on the computer of win7 with memory of 4 GB, and the simulation experiments were implemented in MATLAB R2013. In these simulation experiments, the size of the workspace was set to 300×300. In addition, the locations of the obstacles were initially unknown to the robots. However, the robots knew the search area and the initial locations of the targets. If all of the tasks were completed or the maximum number of iterations $maxItr = 500$ was reached, we ceased execution of the task.

In this section, four different experiments are presented, including the evaluation of robot performance in unknown environments with dynamic obstacles, multi-robot cooperative search, a case of robots failure, and the problem of robot hovering, as well as the performance of comparative testing.

5.1. Unknown Environment with Dynamic Obstacles

In this section, we study the simulation experiments in unknown dynamic environments (dynamic targets, dynamic obstacles). As shown in Figure 4, the blue dots represent the initial locations of the mobile robot, the hollow hexagonal stars represent the initial locations of the targets and a solid hexagonal star indicates that the robot has visited the target position and completed the task. Targets can move randomly in the workspace, where the dotted line represents the trajectory of the moving targets, and the blue hexagonal stars represent the final locations of the moving targets. Figure 4 shows an environment with dynamic obstacles, where the solid black circles represent the initial positions of the obstacles, a hollow circle represents the position of the obstacle after it has moved and the dotted lines are the trajectories of the moving obstacles. The solid lines are the tracks of the robots to the targets. To better illustrate the effectiveness of the algorithm, pause processing was performed in iterations 50 and 90 during the simulation experiment, and the learning rate of the robot was set to $\rho = 1$. The number of iterations is used to reflect the simulation times in this paper. To facilitate the analysis and explanation, the names of some robots, targets and obstacles are marked in the experimental results. The simulation results show that the proposed method can achieve better task assignment and cooperative search in these cases.

As shown in Figure 4, the proposed method achieves better task assignment and cooperative search in an environment with both dynamic obstacles and dynamic targets. As shown in Figure 4a, there are 29 obstacles in the environment, and three of them can move randomly, named O_1, O_2 and O_3. There are eight targets, and three of them can move randomly, named T_1, T_3 and T_5. As shown in Figure 4b, in the 50th iteration, obstacles O_1, O_2 and O_3 and targets T_1, T_3 and T_5 move to their new positions. Each robot can avoid static and dynamic obstacles in an environment with both dynamic obstacles and dynamic targets, which requires that each robot can adjust its path in time according to the real-time location information of the targets and obstacles. The robots need 140 iterations to complete all of the tasks. The simulation results show that regardless of how the target and the obstacles move, the robot can reassign the task and re-plan the path in real time, and it can find a collision-free path to complete the task. It can be seen from the smoothness of the robot's moving trajectories that there is no speed jump problem when the robot moves towards the target. The proposed algorithm has a good robustness.

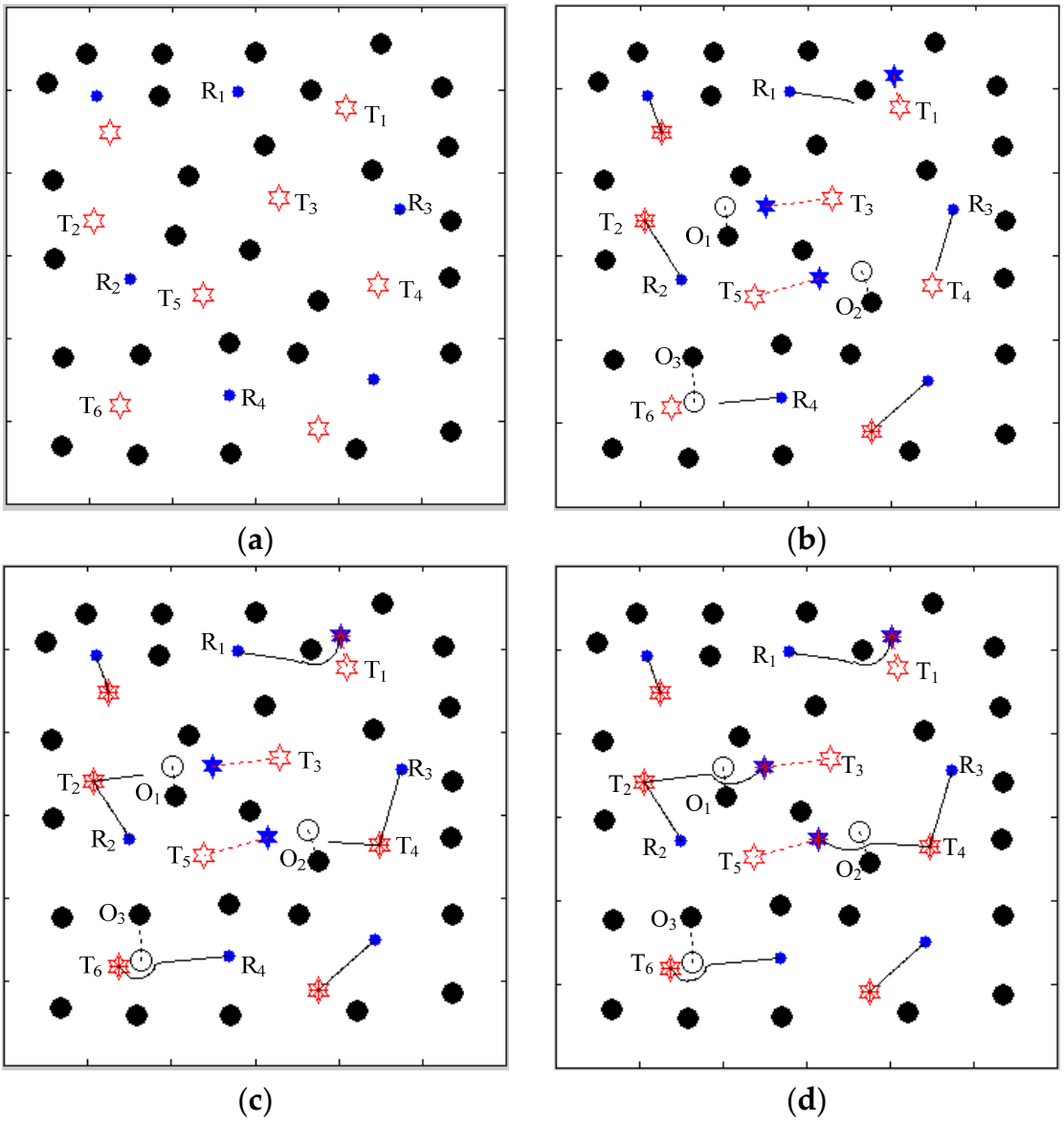

Figure 4. Trajectories of robots searching targets, where some obstacles and some tasks are movable. (**a**) t = 0, (**b**) t = 50, (**c**) t = 90, (**d**) t = 140.

5.2. Multi-Robot Cooperative Search

To better reflect the cooperative search effect of the proposed algorithm, we give the search experimental results in several cases, as shown in Figures 5 and 6. In these experiments, the robot searched according to the proposed DSOM algorithm. The artificial potential field (APF) method is used to replace the velocity $|V_t|$ in the updated weight formula in Equation (15) to update the position of the target point. The target is regarded as the prey in the search process. At this time, the APF does not consider the force between the targets.

To better compare the experimental results of the search, we consider more complex environments in which the number of obstacles increases significantly, as shown in Figures 5 and 6. As shown in Figure 5, the initial positions of the six robots are on the diagonal of the search space, and the target points are on both sides of the diagonal. We take the learning rate ρ as 4, and the coefficient of the target (prey) learning rate as 0.7. However, the neighborhood coefficient at this time is 1.2.

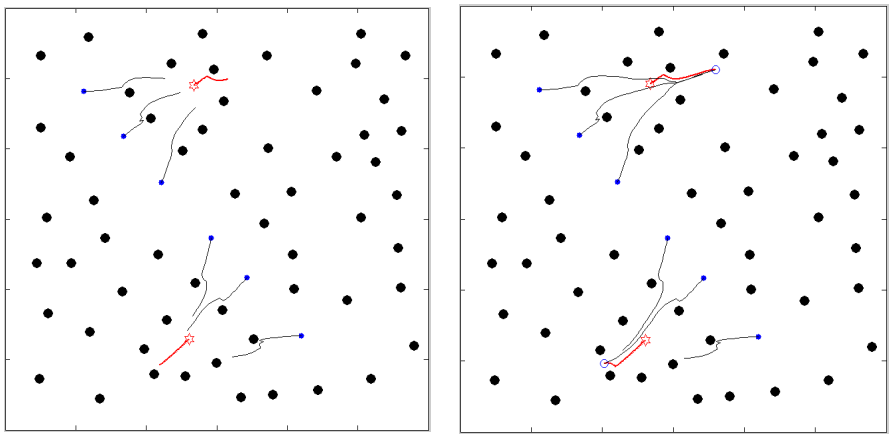

Figure 5. Multi-target search of initial position of multi robot on diagonal. (**a**) t = 20, (**b**) t = 36.

To better see the experimental results, we also pause after 20 iterations, as shown in Figure 5a, and finally complete the target search task after 36 iterations. From the above results, it can be seen that the search tasks of the two targets can be well completed. As shown in Figure 6, there are eight robots and four targets. We take the learning rate of the robot as 3 and the learning rate of the target (prey) as 1, but the neighborhood coefficient is 1.1 at this time. To better observe the experimental results, we also pause after 30 iterations, as shown in Figure 6a. After 61 iterations, we successfully completed the multiple target search task.

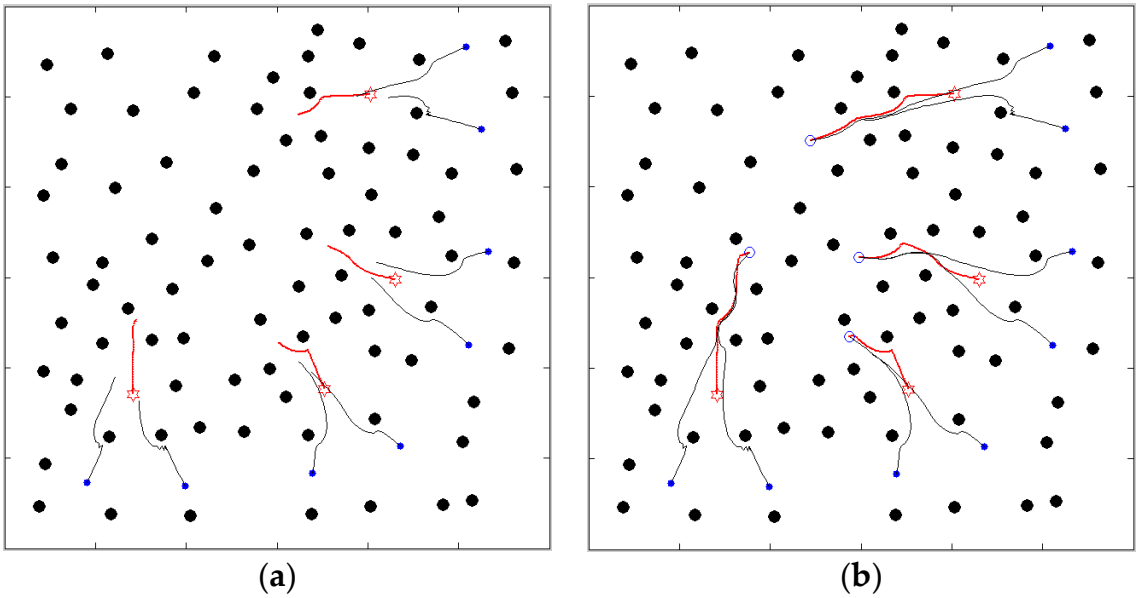

Figure 6. Multi-robot multi-target search in more complex environment. (**a**) t = 30, (**b**) t = 61.

Through the experiment shown in Figures 5 and 6, the effectiveness and efficiency of the proposed DSOM method to complete the multi-target dynamic search are verified. The robot can find the moving target accurately and quickly after locking onto the target according to the locking mechanism and can complete the target search tasks.

5.3. Some Robots Failed

The proposed DSOM presented in this paper has a good robustness. Even in some unpredictable emergencies, the DSOM can still obtain a reasonable solution. In some traditional algorithms,

these unexpected situations often mean fatal errors or system crashes. Figure 7 shows the failure of some robots in the system during the execution of the tasks. Figure 7a shows the initial position of the system. As shown in Figure 7b, the black line represents the trajectory of the robot's movement, and the red cross represents the location where the robot failed. As shown in Figure 7b,c, robots R_4 and R_5 failed on their way to tasks T_2 and T_4, respectively. Additionally, the robots lost their ability to move towards and complete the tasks. In this case, R_3 completes task T_2 after task T_1, and R_6 completes task T_4 after task T_3; finally, all tasks are completed, as shown in Figure 7d. The simulation results show that the proposed method can reassign the tasks and re-plan the path when the robot fails, achieving a good effect.

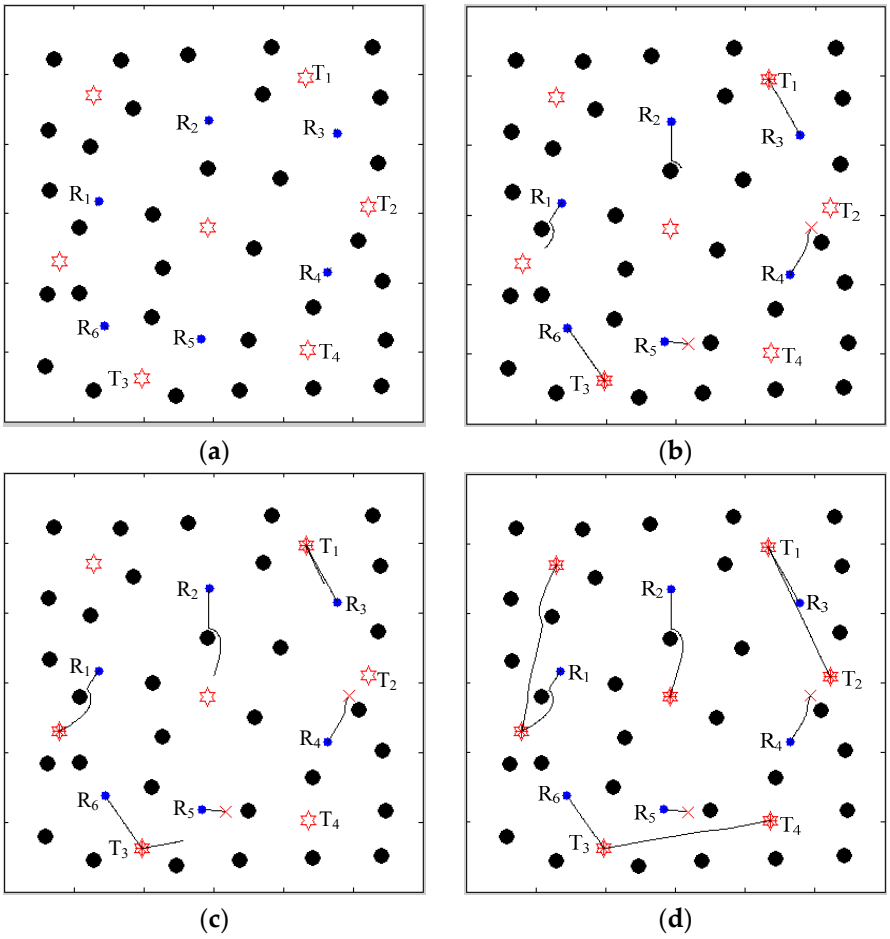

Figure 7. Some robots failed during task execution. (**a**) t = 0, (**b**) t = 50, (**c**) t = 90, (**d**) t = 253.

5.4. Robot Hovering Problem

The proposed method is also capable of addressing the case where the number of tasks is greater than the number of robots. For the case where the number of tasks in the environment is greater than the number of robots (six robots and eight targets), as shown in Figure 8, it is obvious that some of the robots have more than one target to reach. When a robot reaches a target, it should become free and can pursue another target. As seen from the figure, some robots visited only one target, and some robots visited two targets. The main challenge in this case is to choose an optimal visit order for the robots that have more than one target to visit. Due to the competitive effect of the competitive layer in the SOM network, the planned paths for the robots are the shortest paths among all possible routes. As shown in Figure 8d, the proposed method can also adjust the task assignment in real time. Initially, robot R_4 was assigned to complete task T_6, and robot R_5 was assigned to complete task T_5.

However, when robot R_5 completed task T_5, it was closer to task T_6 than robot R_4. At this time, task T_6 is reassigned to robot R_5, and robot R_4 stops working.

The traditional SOM algorithm leads to the robot oscillation and hovering problem. In addition, the robot has no obstacle avoidance function in the traditional SOM method. As shown in Figure 8a, in the traditional SOM neural network task assignment and cooperative search algorithm, robots R_1, R_2, R_4 and R_5 exhibited the oscillation and hovering phenomenon, and the other robots were not assigned tasks. After 500 iterations, the traditional SOM algorithm still failed to progress from this state, while the proposed method only required 172 iterations to complete all tasks and achieve better obstacle avoidance. As shown in Figure 8c, some robots exhibited the oscillation and hovering phenomenon. Robot R_1 completed task T_1, robot R_3 completed task T_3, and then robot R_3 continued to move towards the target. When it moved to the middle of targets T_2 and T_4, according to the traditional winner selection rules, it became the winner of the two tasks, and because of the lack of restrictions on the self-organizing behavior and the random input order, it constantly moves back and forth between the two tasks. At the same time, the hovering phenomenon of robots R_2, R_4 and R_5 made it impossible to complete the task in the end.

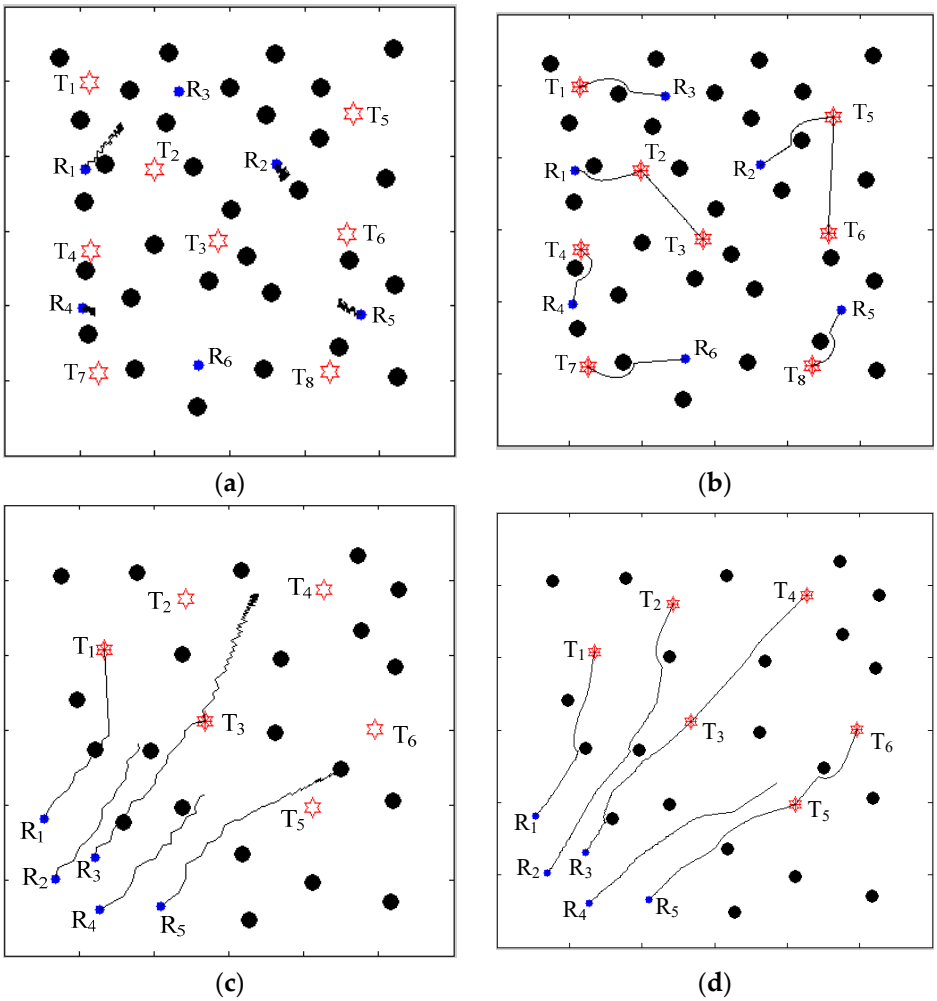

Figure 8. Problem of some robots exhibiting oscillation and hovering. (**a**) The traditional SOM algorithm for the robots' random distribution; (**b**) DSOM for the robots' random distribution. (**c**) The traditional SOM algorithm for the robots' aggregation; (**d**) DSOM algorithm for the robots' aggregation.

The proposed DSOM method solves the problem of obstacle avoidance better and uses a locking mechanism to better address oscillation and hovering, thus improving the success rate of this method. As shown in Figure 8b, robot R_1 completes tasks T_2 and T_3 in turn. In Figure 8b,d, from the motion

trajectory of the robot, it can be seen that the proposed method avoids the oscillation of the robot in the process of task assignment and cooperative search, and from the smoothness of the robot's moving trajectory, there is no problem of a speed jump when the robot moves towards the target. Thus, the entire trajectory is very smooth. In the workspace with obstacles, the improved algorithm can make the robot reach the task position along the shortest path and avoid obstacles automatically when encountered.

5.5. Comparative Testing

To verify the effectiveness and superiority of the proposed method in task assignment and cooperative search under an unknown dynamic environment, comparison studies with the algorithms in reference [1,22] are compared in this section. The algorithm in [1] was proposed for a 3-D underwater environment. To avoid a contingency, thirty different environments with the same size were simulated with the comparison methods in reference [1,22], and the proposed method of DSOM. It should be noted that in our experiments, only the 2-D model of the algorithm in reference [1] is used; the comparison methods in reference [1,22] do not consider the unknown environment, and the location information of obstacles is provided to them as prior information. The success rates and averages of the iterations for each algorithm are reported in Figure 9. There are eight robots in each experiment and these robots and targets are randomly distributed. When the number of targets is more than the number of robots, some robot must visit more than one target. Therefore, the possibility of the oscillation and hovering phenomenon occurring is higher. As shown in Figure 9a, the success rate calculated here refers to the successful experiments that can find all of the targets. When the number of robots is greater than or equal to the number of targets, the success rate of both algorithms is 100%. The larger the number of targets is, the lower the success rate is. The success rate of the DSOM algorithm is better than the comparison methods when the number of targets is greater than 14. In Figure 9b, when the number of targets is less than 10, the number of iterations of the comparison methods in reference [1] and [22] is less than that of the DSOM, but there is not much difference. As a comparison, the number of iterations obtained by using the DSOM is very stable and within a reasonable range. In other words, the larger the number of targets a single robot visits, the more obvious the effectiveness and robustness of the proposed method.

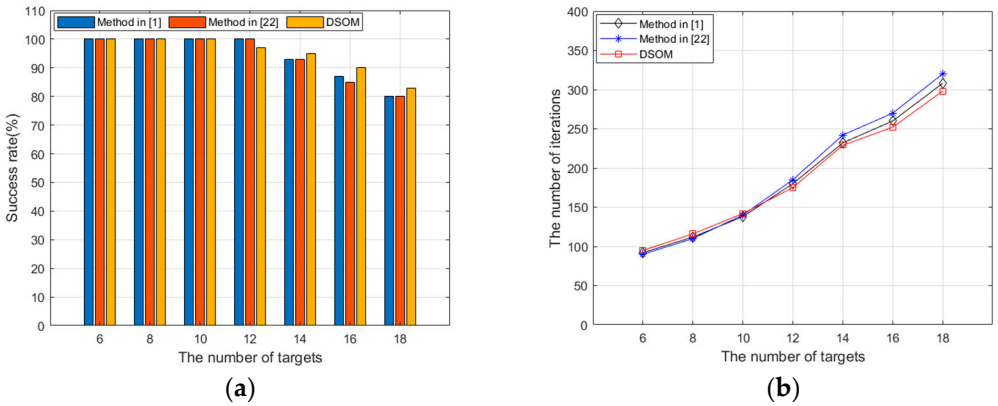

Figure 9. The influence of the number of targets. (**a**) The influence of the number of targets on the success rate, (**b**) the influence of the number of targets on the number of iterations.

To further compare the performance of the comparison methods in reference [1] and [22] and DSOM, simulation experiments are performed for four cases, including a dynamic environment in which the number of targets is equal to that of the robots (Case 1), a dynamic environment in which the number of targets is more than that of the robots (Case 2), a dynamic environment in which some targets are movable (Case3), and a dynamic environment in which some obstacles are movable (Case4). For each case, the numbers and locations of the targets, robots and obstacles are the

same. Thirty experiments are repeated in each case to avoid any contingencies. In each experiment, the number of robots and targets is determined, but the positions of the robots and targets are randomly generated; additionally, the number of iterations and the total path length are recorded. The average number of iterations and the average path length of the proposed method and comparison methods in reference [1,19] are compared in Table 1. Compared with the comparison methods, the proposed method has fewer iterations and a shorter path length in three cases, especially when the number of targets is greater than the number of robots. This difference occurs because the proposed method uses a locking mechanism to solve the problem of oscillation and hovering, and enables the robot to move to the designated target position faster in order to complete the task. Further, the robot uses the DWA to avoid obstacles; the DWA has a better effect on the environment of the unknown obstacle location, and it can better avoid obstacles. Thus, the robot can move to the target position without a collision.

Table 1. Comparisons of the average iteration times and path lengths.

	Average Iteration Times			**Average Path Length**		
	DSOM	**Method in [1]**	**Method in [22]**	**DSOM**	**Method in [1]**	**Method in [22]**
Case 1	156	153	151	288.94	286.81	284.24
Case 2	199	214	205	304.46	314.02	305.32
Case 3	207	220	211	350.88	360.43	353.56
Case 4	187	190	198	342.67	343.62	347.25

The simulation results show that the locking mechanism adopted by the proposed DSOM method improves the robustness of the SOM neural network and reduces the hovering of the robots. Additionally, through the DWA method, the robot can effectively avoid obstacles and access targets in dynamic environments where the location information of the obstacles is unknown, and the robot can avoid speed jumps. Compared with other algorithms, the DSOM has the advantages of requiring less time and utilizing a shorter path length. In addition, the algorithm can address unknown dynamic environments, it can have the number of robots be less than or greater than the target, and it presents cases in which some robots have faults. The DSOM method can also reassign tasks and re-plan paths in real time when the location of the target and obstacle changes. The proposed method can thus better accomplish task assignment and cooperative search.

6. Conclusions

To solve the problems in the task assignment and cooperative search research of multi-robot systems described in this paper, an improved SOM method combined with the adaptive DWA was proposed. The main contributions of the DSOM are as follows. (1) The SOM algorithm is improved by building a locking mechanism on each competing neuron to restrict its self-organizing behavior in order to avoid situations of robot oscillation and hovering. (2) The adaptive DWA is used to update the weights of the SOM's competing neurons for each robot in order to adjust the direction of motion according to environmental information. It can also achieve obstacle avoidance without knowing the location information of the obstacles, optimize the trajectories of robots, and avoid speed jumps. (3) The proposed method can reassign tasks and re-plan paths in real time when the locations of the target and obstacle change. Task assignment and cooperative search in different conditions are well realized in unknown environments in which obstacle location information requires real-time acquisition.

The proposed method of the DSOM was simulated in unknown dynamic environments, as well as in cases where some robots fail and the robot hovering problem exists; this method was also compared with other algorithms. The experimental results show that the proposed method can solve the problems that occur in task assignment and cooperative search in unknown dynamic environments. The DSOM requires fewer iterations, has a better efficiency and a higher success rate in most cases, and has good robustness. In the future, we plan to study task assignment for multi-robot cooperative hunting problems, and we will build a multi-robot experimental platform for testing. The DWA algorithm has

been successfully applied to an ROS robot system; therefore, the proposed method is expected to solve real-world engineering problems and more complex scenarios in the field of intelligent mobile robots.

Author Contributions: H.T. and W.S. gave the conceptualization of the paper and designed the methods; H.T., A.L. and W.S. worked on the simulation results and were involved in the paper writing; H.T. and S.S. analyzed the data and were involved in the paper review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China, grant number U1813205, and the Scientific Research Fund of Hunan Provincial Education Department, grant number 18B435 and 18C0787, and by the Planned Science and Technology Project of Hunan Province, grant number 2016TP1023.

Acknowledgments: The authors would like to thank the Hunan Provincial Key Laboratory of Intelligent Robot Technology in Electronic Manufacturing.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhu, D.; Cao, X.; Sun, B.; Luo, C. Biologically inspired self-organizing map applied to task assignment and path planning of an AUV system. *IEEE Trans. Cogn. Dev. Syst.* **2018**, *10*, 304–313. [CrossRef]
2. Tang, H.; Sun, W.; Yu, H.; Lin, A.; Xue, M.; Song, Y. A novel hybrid algorithm based on PSO and FOA for target searching in unknown environments. *Appl. Intell.* **2019**, *49*, 2603–2622. [CrossRef]
3. Bucolo, M.; Buscarino, A.; Famoso, C.; Fortuna, L.; Frasca, M. Control of imperfect dynamical systems. *Nonlinear Dyn.* **2019**, *98*, 2989–2999. [CrossRef]
4. Tang, H.; Sun, W.; Yu, H.; Lin, A.; Xue, M. A multirobot target searching method based on bat algorithm in unknown environments. *Expert Syst. Appl.* **2020**, *141*, 112945. [CrossRef]
5. Wei, C.; Hindriks, K.V.; Jonker, C.M. Dynamic task allocation for multi-robot search and retrieval tasks. *Appl. Intell.* **2016**, *45*, 1–19. [CrossRef]
6. Liu, S.H.; Zhang, Y.; Wu, H.Y.; Liu, J. Multi-Robot Task Allocation Based on Swarm Intelligence. *J. Jilin Univ. (Eng. Technol. Ed.)* **2010**, *40*, 123–129.
7. Yao, P.; Xie, Z.; Ren, P. Optimal UAV route planning for coverage search of stationary target in river. *IEEE Trans. Control Syst. Technol.* **2019**, *27*, 822–829. [CrossRef]
8. Yao, P.; Cai, Y.; Zhu, Q. Time-optimal trajectory generation for aerial coverage of urban building. *Aerosp. Sci. Technol.* **2019**, *84*, 387–398. [CrossRef]
9. Kohonen, T. Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **1982**, *43*, 59–69. [CrossRef]
10. Xia, B.; Yang, Y.; Zhou, J.; Chen, G.; Lai, Y. Using self organizing maps to achieve lithium-ion battery cells multi-parameter sorting based on principle components analysis. *Energies* **2019**, *12*, 2980. [CrossRef]
11. Ganhadeiro, T.; Christo, E.; Meza, L.; Costa, K.; Souza, D. Evaluation of energy distribution using network data envelopment analysis and kohonen self organizing maps. *Energies* **2018**, *11*, 2677. [CrossRef]
12. Gao, Y.; Sun, Y.; Wang, X.; Chen, F.; Ehsan, A.; Li, H.; Li, H. Multi-objective optimized aggregation of demand side resources based on a self-organizing map clustering algorithm considering a multi-scenario technique. *Energies* **2017**, *10*, 2144. [CrossRef]
13. Zhu, A.; Yang, S.X. A neural network approach to dynamic task assignment of multirobots. *IEEE Trans. Neural Netw.* **2006**, *17*, 1278–1287. [PubMed]
14. Xiang, C.; Zhu, D.; Yang, S.X. Multi-AUV target search based on bioinspired neurodynamics model in 3-D underwater environments. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *27*, 2364.
15. Cao, X.; Zhu, D. Multi-AUV task assignment and path planning with ocean current based on biological inspired self-organizing map and velocity synthesis algorithm. *Intell. Autom. Soft Comput.* **2017**, *23*, 31–39. [CrossRef]
16. Faridi, A.Q.; Sharma, S.; Shukla, A.; Tiwari, R.; Dhar, J. Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment. *Intell. Serv. Robot.* **2018**, *11*, 171–186. [CrossRef]
17. Sun, W.; Lv, Y.; Tang, H.; Xue, M. Mobile robot path planning based on an improved A*Algorithm. *Hunan Daxue Xuebao/J. Hunan Univ. Nat. Sci.* **2017**, *44*, 94–101.

18. Zhao, Y.; Gu, J. Robot path planning based on improved genetic algorithm. In Proceedings of the 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China, 12–14 December 2013; pp. 2515–2522.
19. Brass, P.; Cabrera-Mora, F.; Gasparri, A.; Xiao, J. Multirobot tree and graph exploration. *IEEE Trans. Robot.* **2011**, *27*, 707–717. [CrossRef]
20. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*; Cox, I.J., Wilfong, G.T., Eds.; Springer: New York, NY, USA, 1990; pp. 396–404.
21. Zhang, T.; Zhu, Y.; Song, J. Real-time motion planning for mobile robots by means of artificial potential field method in unknown environment. *Ind. Robot Int. J.* **2010**, *37*, 384–400. [CrossRef]
22. Sun, W.; Tao, Y.; Liu, X.Y.; Xue, M.; Lin, A.; Tang, H. An improved self-organizing map method for task assignment and path planning of multirobot in obstacle environment. In Proceedings of the 2018 Chinese Automation Congress (CAC), Xi'an, China, 30 November–2 December 2018; pp. 7–12.
23. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
24. Hong, Z.; Chun-Long, S.; Zi-Jun, Z.; Wei, A.; De-Qiang, Z.; Jing-Jing, W. A modified dynamic window approach to obstacle avoidance combined with fuzzy logic. In Proceedings of the 2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Guiyang, China, 18–24 August 2015; pp. 523–526.
25. Luo, S.; Kim, J.; Parasuraman, R.; Bae, J.H.; Matson, E.T.; Min, B.-C. Multi-robot rendezvous based on bearing-aided hierarchical tracking of network topology. *Ad Hoc Netw.* **2019**, *86*, 131–143. [CrossRef]
26. Sun, W.; Zhang, F.; Xue, M.; Hu, W.; Li, L. An SOM-based algorithm with locking mechanism for task assignment. In Proceedings of the 2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), Ningbo, China, 19–21 November 2017; pp. 36–41.