International Journal of Robotics and Automation, Vol. 21, No. 2, 2006

A SOM-BASED MULTI-AGENT ARCHITECTURE FOR MULTIROBOT SYSTEMS

A. Zhu* and S.X. Yang*,**

Abstract

In this paper, a self-organizing map (SOM)-based multi-agent architecture is proposed for multirobot systems. It is capable of controlling a group of mobile robots to complete multiple tasks simultaneously. By cooperative and competitive behaviours, the group of mobile robots can automatically arrange the total task, and dynamically adjust their motion whenever the environment is changed. As an implementation, it can control a group of mobile robots to complete multiple tasks at different locations, such that the desired number of robots will arrive at every target location from arbitrary initial locations. The proposed approach integrates the task requirement of robots and the robot motion planning, such that the robots can start to move before their destinations are finalized. The robot navigation can be dynamically adjusted to guarantee that each target location has the desired number of robots, even under unexpected uncertainties, such as when some robots break down, some robots and/or some tasks are added, or some tasks are changed. Unlike most conventional models that are suitable to static environments only, the proposed approach is capable of dealing with changing environments. In addition, the proposed algorithm can be applied to the path planning of multirobot systems, where a group of robots is coordinated to visit a set of depots. The effectiveness and efficiency of the proposed approach are demonstrated by simulation studies.

Key Words

Multi-agent, multirobot systems, task assignment, cooperation, competition, self-organizing map (SOM)

1. Introduction

Multirobot systems have been the subject of much research since the 1970s for a wide variety of tasks in both multimanipulation and multimobile robot systems [1–6]. Recently there has been a large range of task domains for multirobot systems, such as forging, multitarget observation, box pushing, exploration, path planning, and soccer. Balch [7] proposed an unaware, reactive multirobot system, composed of behaviour-based robotic agents, to deal with a multiforaging task, in which two types of objects must be collected and delivered to different locations depending on their type. Parker [8] proposed a reactive, distributed, strongly coordinated multirobot system to deal with the multitarget observation. The task consists of maximizing the time during which each of the moving targets is being observed by at least one of the robotic agents. Kube and Bonabeau [9] introduced a reactive, unaware multirobot system to deal with a box-pushing task. This system is a classical example of the swarm approach. The robotic agents are homogeneous and rely on a behaviour-based architecture. The authors implemented a set of behaviours inspired by the way some ant species organize themselves to accomplish a pushing task. Hughes [10] proposed a distributed, reactive multirobot system to deal with an exploration task in an office-like environment. The system is designed to study the problem of grounding representations and communications of the robotic agents in real perceptions. Robotic soccer is one of the tasks that has been considered an interesting test bed for research in multi-agent and multirobot cooperation [11].

Other interesting contributions to the work on multirobot systems are the surveys by Cao *et al.* [12], Dudek *et al.* [13], and Iocchi *et al.* [6]. In [12] several methods for characterizing a multirobot system are discussed. A classification of multirobot system is discussed in [13]. It focuses on the communication and computation aspects. In [6] multirobot systems are classified by cooperative capabilities.

In many applications, a multirobot system is required to complete some complex or heavy tasks faster and more efficiently than a single-robot system. For example, several manipulation robots could cooperate for some complicated assembly tasks in manufacturing; a robot team could complete an assigned task rapidly and efficiently by separating the overall task into several subtasks and then executing the subtasks simultaneously. Multirobot systems can improve performance and reliability. However, in a multirobot system the most challenging issue is the coordination and cooperation of these robots to satisfactorily perform the overall task. The multi-agent system gives us a good way to solve the problem.

* School of Engineering, University of Guelph, Guelph, Ontario, N1G 2W1, Canada; e-mail: {azhu, syang}@uoguelph.ca
** Corresponding author

(paper no. 206-2793)

The multi-agent system is an emerging subfield of artificial intelligence (AI). It tries to provide principles for construction of complex systems, involving multiple agents and mechanisms for coordination of independent agents' behaviours. Although there is no generally accepted definition of "agent" in AI, usually an agent system is defined as a computational system that tries to fulfill a set of goals in a complex, dynamic environment. For example, an individual robot with goals, actions, and domain knowledge, situated in an environment, is considered an agent in a multirobot system. Then the total system is considered a multi-agent system. The individual agents (robots) can study self-organization between each other, coordinate actions, and accomplish the task. Another example is a mobile robot system with two different wheels. Each wheel is considered to be an individual agent, which can always select its velocity according to its measurement of the distance to obstacles. Multi-agent systems have been studied by several research groups in recent years. Details can be found from the survey in [14–16]. In [14], a series of general multi-agent scenarios is presented, including the simplest multi-agent scenario, homogeneous noncommunicating agents, the full range of possible multi-agent systems, and highly heterogeneous communicating agents. Those scenarios are biased towards one's that use machine learning approaches. From the robot soccer point of view, different types of agents and different structures of the agent cooperation for multi-agent systems are discussed in [15]. In [16] communication between agents as an important component of multi-agent systems is briefly discussed.

In multi-agent systems, the basic problem is to determine what action an agent should take in a given situation. In particular, a dynamic environment with competition between agents makes the problem more difficult and complex [15]. In this paper, a simple SOM-based multi-agent architecture is proposed for multirobot systems. It is capable of controlling a group of mobile robots to accomplish multiple tasks, which cannot be completed by a single robot. The group of mobile robots can automatically arrange the total task and dynamically adjust their motion whenever the environment is changed, such as when some robots break down, some robots and/or some tasks are added, or some tasks are changed.

This paper is organized as follows. As a test bed for the proposed multi-agent architecture, a definition of task assignment of multirobot systems is given in Section 2. Section 3 presents the proposed architecture. The application of the proposed architecture to the path planning of multirobot systems is given in Section 4. Section 5 provides the simulation results. Some concluding remarks are given in Section 6.

2. Problem Statement

In multirobot systems, the main challenge is the cooperation and competition of the robots in performing a single task. The task assignment problem is an important issue in multirobot systems. It allows for the coordination and cooperation of a multirobot system. In this paper, assume that several autonomous mobile robots are randomly distributed in a bounded area and that a set of targets is also randomly distributed in this workspace. Every target requires a specific number of robots to complete a task at each location. The goal is to dynamically assign a team of robots to every target location with a minimal or near-minimal total cost. For every robot, the cost is evaluated by the travel distance from its initial to final location. The total cost is defined as the sum of all of the costs from each robot. The task is completed when every target has the desired number of robots reached. An example workspace with the initial robots and the target locations is shown in Fig. 1, where the dots represent initial mobile robot locations and the squares represent the target locations. In addition, it is assumed that the robots are homogeneous mobile robots with the basic capabilities for navigation, obstacle avoidance, communication, and location recognition. This paper covers only how to assign the robots to the targets and how to control the motion of mobile robots.

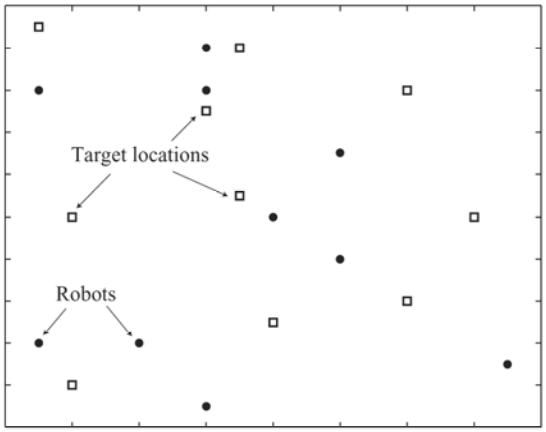

Figure 1. A workspace with multiple robots and target locations.

Some studies have found efficient solutions for the task of controlling a group of robots to move to target locations, but most algorithms are proposed for the task assignment problem in static environments, for example, the graph matching algorithm [17], distributed auction algorithm [18], genetic-based algorithm [19], and agent-based algorithm [20]. These algorithms mainly focus on the target assignment problem without considering the motion planning of robots. The consequence is that the robots cannot move until their destinations are finalized. In addition, these approaches cannot handle the situation with movable targets.

The task assignment of multirobot systems in this paper emphasizes not only the assignment of the desired number of robots to every target location, but also the reduction of the total robot travelling distance from their initial to target locations, where the targets can be either static or movable.

3. The Proposed Architecture

In this section, a SOM-based multi-agent architecture is proposed. The details for solving the task assignment

problem of multirobot systems using the proposed multi-agent architecture are then given.

3.1 The Architecture of the Proposed Model

Inspired by the ubiquity of cortical maps in the central nervous system, a SOM-based network [21] we designed that has two layers, as shown in Fig. 2. The input layer has two nodes (x_i, y_i) and the output layer has K nodes $(R_1, R_2, \ldots, R_K)$. Every node of output layer is fully connected to the nodes of the input layer. For example, the j-th node of output layer R_j has the connection vector (w_{jx}, w_{jy}). The connection strengths of the output layer nodes with the input layer nodes are given by the 2D vector $\{w_{jx}, w_{jy}\}$, where $j = 1, 2, \ldots, K$.

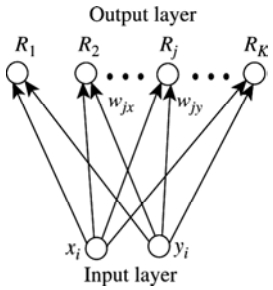

Figure 2. The structure of the SOM-based network.

The SOM has self-organizing behaviour in that the output layer nodes tend to attain connection vectors that capture characteristics of the input vector space. It is capable of lacing the associated outputs of similar inputs close to each other; the more similarity in the input features, the closer the location in the output map. In order to achieve a satisfactory performance of an overall task of multirobot systems, including solving the task assignment problem of multirobot systems, a SOM-based multi-agent architecture is proposed as follows.

Unlike in some multi-agent models, where subfunctions of a robot system are defined as agents [22, 23], in this proposed multi-agent architecture every robot is considered an agent. In the proposed architecture, the input nodes represent the tasks and the output nodes represent the robots. The connection vector represents the actions of the robots. According to the concept of multi-agents, the output nodes can be considered individual agents, which means every robot is considered an agent. The agents determinate what action (connection vector modification) should be taken in a given situation (input task) by cooperation and competition between the agents.

First, the proposed SOM-based multi-agent architecture has competitive behaviour. According to a task represented by the input nodes, the agents represented by the output nodes compete to be the winner, which has the priority to perform the task. In order to restrict a node to be the winner for more than one time in each epoch, an inhibitory index is defined for each node, which puts the winner node aside from the future competition, providing more opportunity for other nodes. The winner node is considered the node with the minimum distance towards the input node.

Second, the proposed SOM-based multi-agent architecture has cooperative behaviour. After an agent becomes the winner, the system is to decide the neighbour agents of the winner agent. The neighbourhood function determines the influence (the attraction strength) of the input data on the winner and the neighbouring agents. The attraction force should be highest for the winner agent, diminish as the proximity of the agent to the winner agent decreases, and have no effect on the agents far away from the winner agent.

Third, the winner and the neighbour agents perform the task by modifying their connection vectors, and the other agents stay without changes. It is obvious that the modification of the connection vectors should depend not only on the original relationship between the winner agent with its neighbour agents and the input task, but also on the neighbourhood function and the network learning rate.

A flow diagram of the proposed algorithm for multirobot systems is shown in Fig. 3.

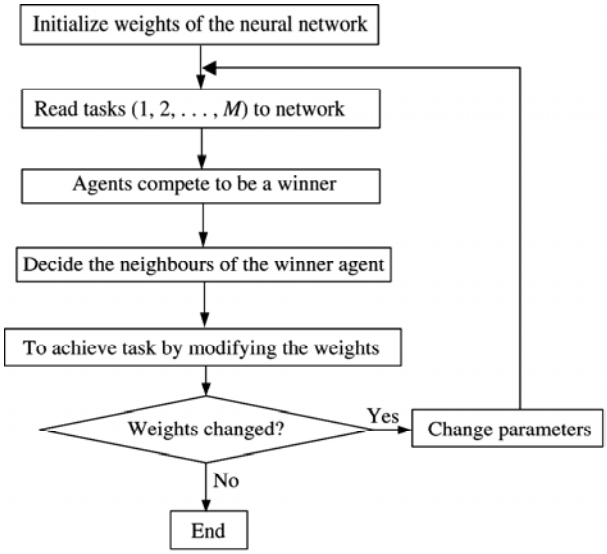

Figure 3. The flow diagram of the proposed algorithm for multirobot systems.

3.2 Task Assignment of Multirobot Systems

Assume there are K robots and M targets in the workspace. The input layer nodes (x_i, y_i) represent the Cartesian coordinate of the i-th target T_i in the workspace. The output layer nodes denoted by $(R_1, R_2, \ldots, R_K)$ represent the coordinates of the robots considered individual agents. The connection vector (w_{jx}, w_{jy}) represents the coordinate of the j-th agent after its action for the input task. Fig. 4 shows a geometrical representation of the proposed architecture for the task assignment of multirobot systems, where the solid square represents the coordinate of the i-th input task, the dots represent the positions of the robot agents $(w_{jx}, w_{jy}), j = 1, 2, \ldots, K$, and the empty squares represent the next inputs in one epoch of all input task datasets.

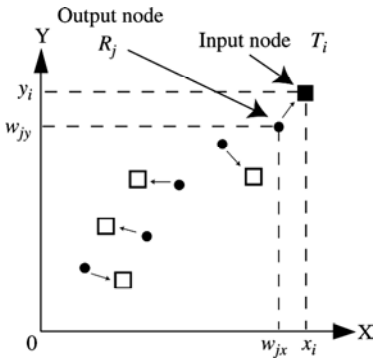

Figure 4. The geometrical representation of the proposed architecture to the task assignment of multirobot systems in an epoch.

The vectors $\{w_{jx}, w_{jy}\}, j = 1, 2, \ldots, K$ are initialized as the coordinates of initial locations of the K robots. The input task datasets are inserted into the network one by one in a random order. For a given task location $T_i = (x_i, y_i)$ as input, the robot agents (output nodes) compete to be the winner by the following rule:

$$J_i \Leftarrow D_{J_i} = \min\{D_{ij}, j = 1, \ldots, K; \quad \text{and} \quad j \in \Omega\} \quad (1)$$

with D_{ij} defined as:

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

where $i = 1, 2, \ldots, M$; D_{ij} is the distance between the j-th agent node and the i-th input task node in the coordinate system; D_{J_i} is the minimal distance of D_{ij}; J_i represents the node index of the winner agent in the output layer to the input task node $T_i = (x_i, y_i)$, which is the coordinate of the i-th target location; $R_j = (w_{jx}, w_{jy})$ is the Cartesian coordinate of the j-th agent as an output node; $|\cdot|$ denotes the Euclidean distance; and Ω is the set of nodes that have not been the winner yet during an epoch.

The action of the agent is described as:

$$R_j(t+1) = T_i(t), \quad \text{if } D_{ij} < \eta D_{\min} \quad (3)$$

Otherwise:

$$R_j(t+1) = R_j(t) + \beta f(d_j)(T_i(t) - R_j(t)) \quad (4)$$

where β is the learning rate, η is a small constant (normally less than 0.5), and $D_{\min}$ represents the minimum distance of any two nodes of targets. The introduction of $D_{\min}$ can obviously reduce the computing time of the algorithm. The neighbourhood function $f(\cdot)$ is defined as:

$$f(d_j) = \begin{cases} e^{-d_j^2/G^2}, & \text{if } d_j < r \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $d_j = |R_j - R_{J_i}|, j = 1, 2, \ldots, K$, $R_{J_i} = \{w_{J_{ix}}, w_{J_{iy}}\}$ is the Cartesian coordinate of the winner agent; $G = (1-\alpha)^t G_0$ is a nonlinear function (G_0 is initialized as 10); t is the number of epochs; α is the change rate that determines the computation time (the smaller the α is, the longer the computation time is; the smaller the α is, the shorter the total path of the robots is (normally in the range [0.05, 0.01])); and r is a small constant (normally less than 0.4) indicating the range of neighbourhood.

4. Path Planning of Multirobot Systems

The proposed multi-agent architecture can also be applied to the path planning of multirobot systems. Assume that a group of mobile robots is randomly distributed in a bounded area and the destinations of the robots are given randomly. There are a set of depots also randomly distributed in this workspace. The goods in the depots have to be carried to one of the destinations. Fig. 5 shows the initial situation with the robots and depots. Assume that the robots are homogeneous mobile robots with the basic capabilities for navigation, obstacle avoidance, object recognition, and object handling. The goods can be carried away when the depot is visited by any robot. Each robot starts from its start position, visits several depots to carry the goods away, and then reaches its destination. When all of the goods are carried away and every robot reaches its destination, the whole system is determined to have completed its goal. The cost of a robot is the travelling distance from its start position, through some depots, to its destination position. In order to finish the task in optimal ways, it is necessary to plan the path of each mobile robot in this workspace. The objectives of the path planning are to minimize the total travelling distance of the robots and assign average number of depots to every robot.

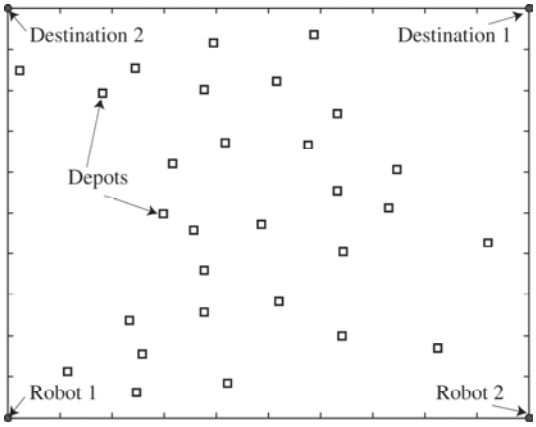

Figure 5. Workspace with robots, including the start and destination positions and depots.

There are many studies on efficient solutions to path planning for a single robot using global method, potential field method, hybrid approach, neural network, neuron-fuzzy network, or genetic algorithm (e.g., [24–26]), which mainly focus on obstacle avoidance.

In this paper, the proposed multi-agent architecture is used in the path planning of multirobot systems, which focuses on the coordination of multirobots and near-optimal solutions within a reasonable computation time. It emphasizes the reduction of the total robot travelling cost and a workload balance for every robot. Similar to the

situation of the task assignment of multirobot systems, assume that K robots and M depots are randomly distributed in a workspace. Therefore, a SOM network is designed as shown in Fig. 2. The input layer nodes (x_i, y_i) represent the Cartesian coordinate of the i-th depot T_i in the workspace. The output layer has KM nodes $(R_{11}, \ldots, R_{1M}, R_{21}, \ldots, R_{2M}, \ldots, R_{K1}, \ldots, R_{KM})$. When the process of path planning is completed, the M depots attract M nodes from KM output layer nodes to form K paths for the K robots. Every robot has its own path, including the start position, several depots, and its destination position. The total path of the robots will visit all of the M depots. The order of M nodes distributed in the K lines is the objective of the path planning. The geometrical representation of the proposed architecture to the path planning of multirobot systems is shown in Fig. 6 with only one of the K robots, its start and destination position, a virtual line with M virtual notes on it, and several depots as input inserted one by one during an epoch.

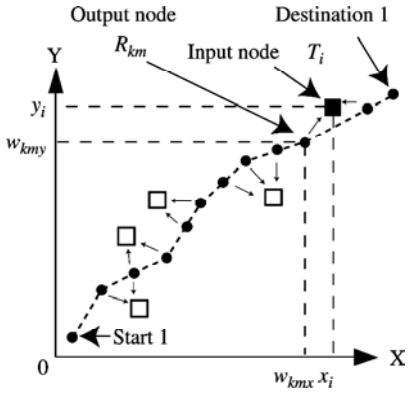

Figure 6. The geometrical representation of the proposed architecture to the path planning of multirobot systems.

At the beginning, the vectors (w_{kmx}, w_{kmy}) are initialized to form K virtual lines for K robots. Every robot has a virtual line from its start position to its destination with M virtual nodes on it. The M connection vectors are initialized as points averagely distributed on the virtual line connecting the start and the destination position of the robot. After that, the input data are inserted sequentially into the network. For a given depot location $T_i = (x_i, y_i)$ as input, the output nodes compete to be the winner according to a specified criterion described as:

$$[N_k, N_m] \Leftarrow \min\{D_{ikm},\ i = 1, \ldots, M; \\ k = 1, \ldots, K;\ m = 1, \ldots, M;\ \text{and}\ \{k, m\} \in \Omega\} \quad (6)$$

where $[N_k, N_m]$ means the N_m-th node on the N_k-th virtual line is the winner; Ω is the set of nodes that have not been the winner yet in an epoch; and D_{ikm} is given as:

$$D_{ikm} = |T_i - R_{km}|[1 + P] \quad (7)$$

where:

$$|T_i - R_{km}| = \sqrt{(x_i - w_{kmx})^2 + (y_i - w_{kmy})^2} \quad (8)$$

Here $|\cdot|$ and $T_i = (x_i, y_i)$ have the same meanings as in (2); $R_{km} = (w_{kmx}, w_{kmy})$ is the coordinate of the m-th node on the k-th virtual line, $k = 1, 2, \ldots, K, m = 1, 2, \ldots, M$; and the term P controls the equitable distribution of workload for every robot, which is defined as:

$$P = \frac{L_k - V}{1 + V} \quad (9)$$

where L_k is the path length of the k-th robot, $k = 1, 2, \ldots, K$; and V is the average path length of the robots. The winner is considered not only the node with the minimum distance towards the input data, but also the node in the virtual line with a lower workload.

The neighbouring nodes of the winner are decided according to the neighbourhood function $f(d_j)$, which is the same as in (5). But here $d_j = ||j - N_m||$, $j = 1, 2, \ldots, M$, which is the distance between the j-th node and the winner N_m on the N_k virtual line. $\|\cdot\|$ denotes the absolute value. The rule of updating the connection vectors is the same as in (3) and (4).

5. Simulation Studies

In order to gain a better understanding of the proposed SOM-based multi-agent architecture and its implementation in the task assignment and path planning of multirobot systems, several different simulation cases are studied in this section. The algorithms are coded in MATLAB and implemented on a 730 MHz Pentium III PC with 260 MB RAM and Windows NT.

5.1 Task Assignment of Multirobots

To illustrate the effectiveness of the proposed approach to the task assignment of multirobot systems, four different cases are studied in this subsection.

In the first case, the number of robots is equal to the number of targets. The time-dependent process of a multirobot system is illustrated by four time steps in Fig. 7. At the initial state (Fig. 7(a)), ten targets are randomly

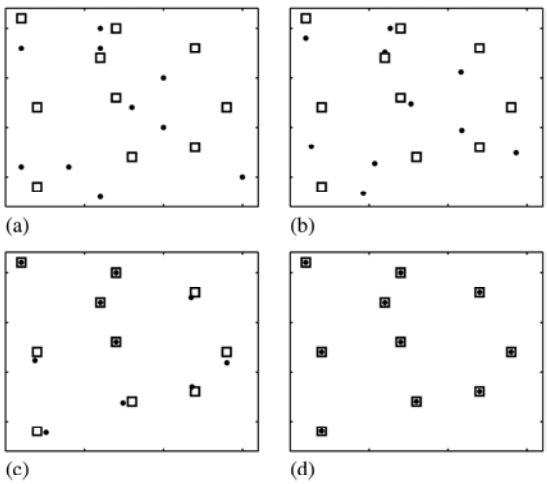

Figure 7. The evolution of the solution at the initial (a), after 5 (b), 20 (c) epochs, and at the final (d) status.

distributed in the area represented by ten squares, and ten mobile robots are randomly distributed represented by ten dots. The evolution after 5 (Fig. 7(b)) and 20 (Fig. 7(c)) epochs shows that the robots move to different targets gradually and respectively. At the end, a stable state (Fig. 7(d)) emerges while every target is occupied by a robot. The whole process of all time steps shows the tracks of the robots to the targets in Fig. 8. The proposed approach combines the task requirement of robots and the robot motion planning together, such that the robots can start to move before their destinations are finalized.

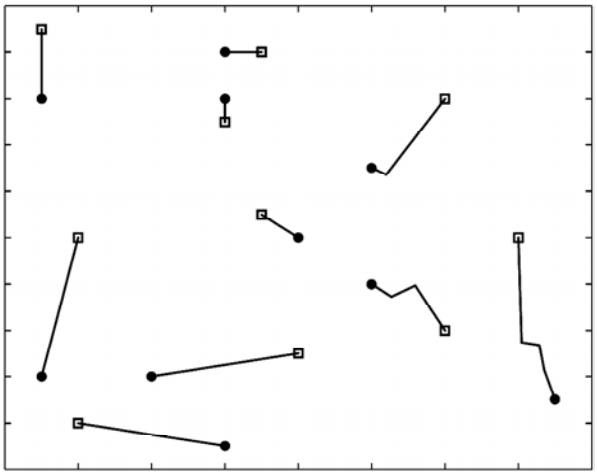

Figure 8. The tracks of the robots to the target locations, where the number of robots and the number of targets are the same.

The second case shows that the proposed approach can deal with a complicated case that cannot be handled by conventional approaches, in which some targets need more than one robot. Fig. 9 shows that target T_1 needs two robots, T_2 needs one, T_3 needs two, and T_4 needs three. At the beginning, robots R_1 and R_9 move to target T_1, while R_2 moves to T_2. After several steps, R_1 and R_2 move to the T_1 and R_9 stops without moving any more. Robot R_{10} does not move a step, because it is relatively far away from any target.

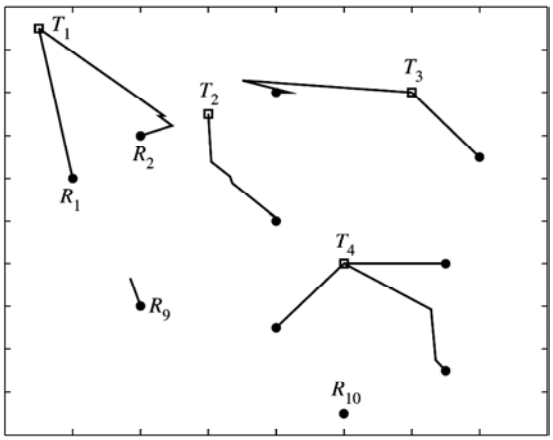

Figure 9. The tracks of the robots to the targets, where targets T_1, T_3 and T_4 need more than one robot there.

The third case shows that the proposed approach is an error-resistant system. It works in the case of an unexpected event. This error resistivity of the cooperation process allows for a sudden change of the environment, such as the breakdown of some robots during the movement. Fig. 10 shows this kind of situation. At the beginning period, robot R_1 moves to target T_1, R_2 moves to T_2 and away from T_1, R_4 moves to T_3, and R_5 moves to T_4 and away from T_3. Assume that robots R_1 and R_4 break down, shown as multiplication signs in Fig. 10, after a certain time. In this situation, R_2 returns and moves to T_1, and R_5 returns and moves to T_3. At the end, every target location is occupied by a robot by taking advantage of the multi-agent cooperation behaviour of the proposed approach.

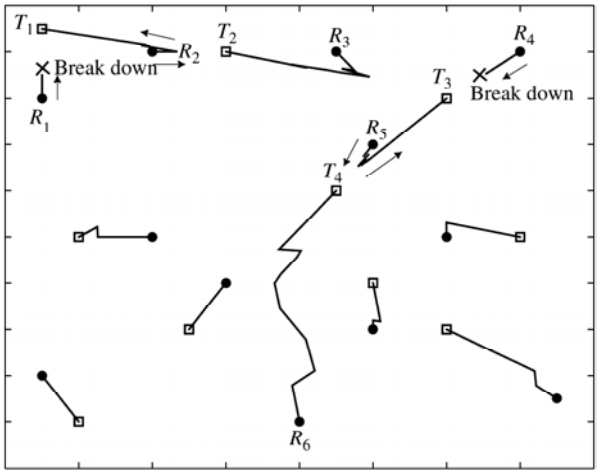

Figure 10. The tracks of the robots to the targets, where robots R_1 and R_4 break down on the way towards targets.

In the fourth case, the targets are assumed to move randomly. Fig. 11 shows this kind of situation. The targets $T_1, \ldots, T_4$ move randomly in the workspace. The robots $R_1, \ldots, R_4$ can catch the respective targets. The advantage of the proposed method is obvious: it has the cooperation feature to deal with dynamic environments.

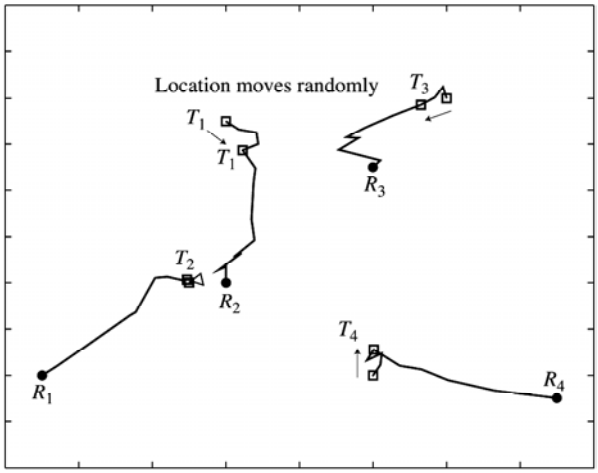

Figure 11. The tracks of the robots to moving targets.

5.2 Path Planning of Multirobots

To illustrate the effectiveness of the proposed approach to the path planning of multirobot systems, this algorithm is applied to four different cases, including robots at the same start position that return to the start position, robots at the same start position that go to different destinations, the start position of a robot being the destination of other robot, and so on.

In the first case, the proposed approach is applied in a workspace with three robots having the same start position and returning to this position. Assume that the start position S is located anywhere in the workspace as shown in Fig. 12(a). At the initial situation, the vectors (w_{kmx}, w_{kmy}) are set to the coordinate of the point S. During the processing, three paths, which form three rings, gradually stretch towards the depots until they reach every depot. Fig. 12 shows the status of the workspace at initial (Fig. 12(a)), after 5 (Fig. 12(b)) and 100 (Fig. 12(c)) epochs, and at the final (Fig. 12(d)) situation. The result shows that every depot is visited by only one robot, the workload is averagely distributed to the robots, and the total path length is near optimal. According to our experience, the computational time is very short, within only six to seven seconds.

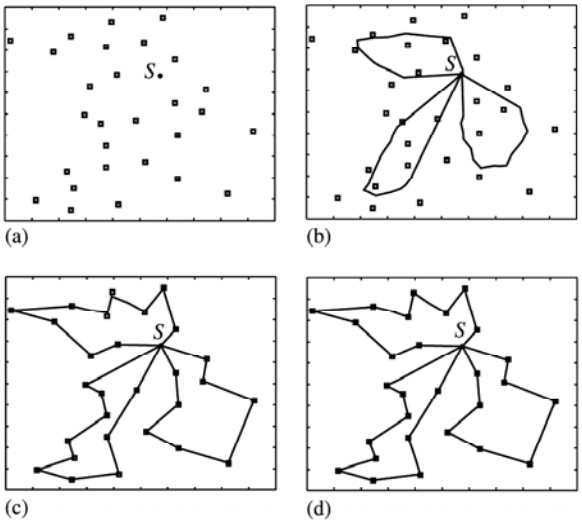

Figure 12. The evolution of the solution for three robots with the same start and destination position at S at the initial (a), after 5 (b), 100 (c) epochs, and at the final (d) status.

In the second case, the proposed approach is applied in a workspace with four robots having the same start position but different destinations. Assume that the start position S is located in the central area of the workspace, and the destinations D_1, D_2, D_3, and D_4 are located at the four corners of the workspace as shown in Fig. 13(a). At the initial situation, four virtual lines with M points averagely distributed on every line from the S to D_i $(i = 1, 2, 3, 4)$ are formed. During the processing, the four virtual lines gradually stretch towards the depots until they reach every depot to form four paths. Fig. 13 shows the status of the workspace at initial (Fig. 13(a)), after 5 (Fig. 13(b)) and 100 (Fig. 13(c)) epochs, and at the final (Fig. 13(d)) situation.

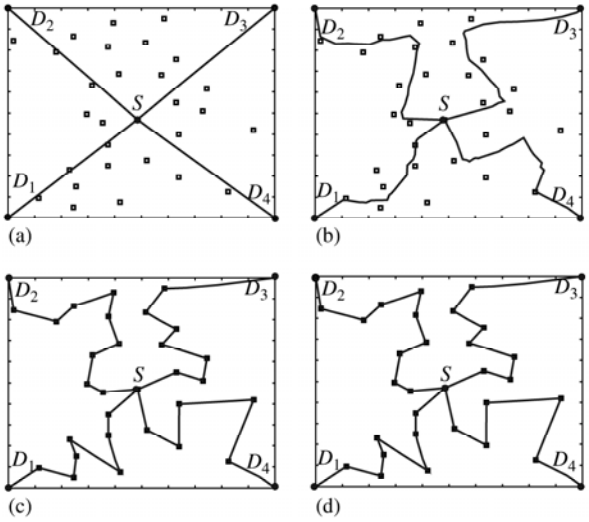

Figure 13. The evolution of the solution for four robots at the initial (a), after 5 (b), 100 (c) epochs, and at the final (d) status, where the start positions are same at S, and the destinations are at D_1, D_2, D_3, and D_4 respectively.

In the third case, there are two robots in the workspace. The start position of a robot is the destination of another robot. Assume that the start position S_1 of robot 1 and the destination D_2 of robot 2 are the same and are located in the lower left side of the workspace, and that the start position S_2 of robot 2 and the destination D_1 of robot 1 are the same and are located in the upper right side of the workspace as shown in Fig. 14(a). At the initial situation, two virtual lines overlapping from the start to destination positions of the robots are formed. Every virtual line has M points distributed on it averagely. During the processing,

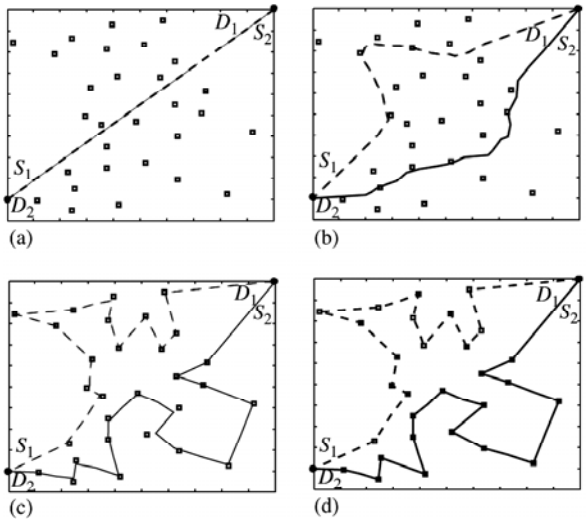

Figure 14. The evolution of the solution for two robots at the initial (a), after 5 (b), 100 (c) epochs, and at the final (d) status, where S_1 and D_2 are at the same location, and so are S_2 and D_1.

the two virtual lines gradually stretch towards the depots until they reach every depot to become two paths. Fig. 14 shows the status of the workspace at initial (Fig. 14(a)), after 5 (Fig. 14(b)) and 100 (Fig. 14(c)) epochs, and at the final (Fig. 14(d)) situation.

In the last case, there are two robots in the workspace. The start position S_1 of robot 1 is located in the lower left side of the workspace, and its destination D_1 is located in the upper right side of the workspace. The start position S_2 of robot 2 is located in the upper left side of the workspace, and its destination D_2 is located in the lower right side of the workspace. At the initial situation, two virtual lines from the start to destination positions of the robots are formed. Every virtual line has M points distributed on it averagely. During the processing, the two virtual lines gradually stretch towards the depots until they reach every depot to form two paths. Fig. 15 shows the status of the workspace at initial (Fig. 15(a)), after 5 (Fig. 15(b)) and 100 (Fig. 15(c)) epochs, and at the final (Fig. 15(d)) situation.

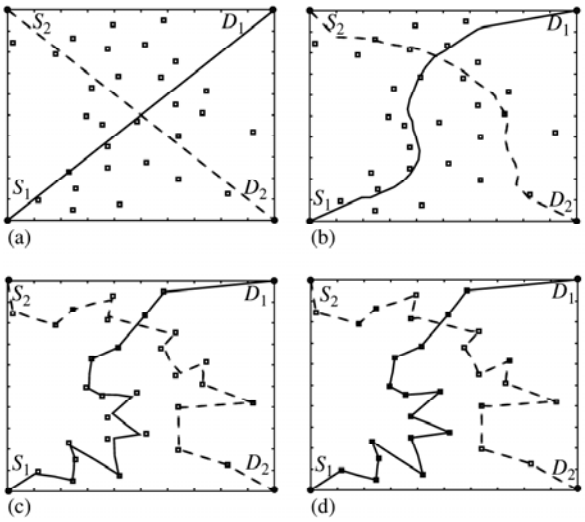

Figure 15. The evolution of the solution for two robots with arbitrary start and destination positions at the initial (a), after 5 (b), 100 (c) epochs, and at the final (d) status.

6. Conclusion

The proposed SOM-based multi-agent architecture carries out a simple method for multirobot systems to complete multiple tasks, which a single robot can not accomplish. With the competition and cooperation process, the proposed approach has several interesting features and advantages. For the task assignment of multirobot systems, the model combines target assignment and motion planning together, which allows the robots to start moving before their destinations are finalized. It can deal with a sudden change of the situation, such as the breakdown of some mobile robots. It can also deal with some complicated cases, such as assigning more than one robot to a target location. In addition, it is capable of dealing with a changing environment, such as the targets being movable. Furthermore, the proposed method can be extended to deal with the path planning problem of multirobot systems, which focuses on the coordination of multirobots and the near-optimal solutions with a reasonable computation time for a group of robots to visit a set of depots.

Acknowledgement

This work was supported by Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] E. Nakano, S. Ozaki, T. Ishida, & I. Kato, Cooperational control of the anthropomorphous manipulator "melarm," *Proc. 4th Int. Symp. Industry Robot*, Tokyo, 1974, 251–260.

[2] S. Fujii & S. Kurono, Coordinated computer control of a pair of manipulators, *Proc. 4th IFToMM World Congress*, Newcastle-upon-Tyne, UK, 1975, 411–417.

[3] S. Hayati, Hybrid position/force control of multi-arm cooperating robots, *IEEE Int. Conf. Robot Automation*, San Francisco, CA, 1986, 82–89.

[4] I.D. Walker, R.A. Freeman, & S.I. Marcus, Analysis of motion and internal force loading of objects grasped by multiple cooperating manipulators, *International Journal of Robot Research, 10*, 1991, 396–409.

[5] B.J. Diressen, J.T. Feddema, & K.S. Kwok, Decentralized fuzzy control of multiple nonholonomic vehicles, *Journal of Intelligent and Robotic Systems*, *26*, 1999, 65–78.

[6] L. Iocchi, D. Nardi, & M. Salerno, *Balancing reactivity and social deliberation in multi-agent systems: From RoboCup to real-world applications, reactivity and deliberation: A survey on multirobot systems* (Berlin: Springer-Verlag, 2001).

[7] T. Balch, The impact of diversity on performance in multi-robot foraging, *Proc. Association for Computing Machinery International Conference on Autonomous Agents*, Seattle, WA, May 1999, 92–99.

[8] L.E. Parker & B.A. Emmons, Cooperative multirobot observation of multiple moving targets, *Proc. Int. Conf. on Robotics and Automation*, vol. 3, Albuquerque, NM, April 1997, 2082–2089.

[9] C.R. Kube & E. Bonabeau, Cooperative transport by ants and robots, *Robotics and Autonomous Systems*, *30*(1–2), 2000, 85–101.

[10] L. Hughes, Grounded representations for a robots team, *Proc. 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, Takamatsu, Japan, November 2000, 2248–2253.

[11] P. Stone *et al.*, Overview of RoboCup-2000, in *RoboCup 2000: Robot Soccer World Cup IV*, Melbourne, 2000, 1–28.

[12] Y.U. Cao, A.S. Fukunaga, & A. Kahng, Cooperative mobile robotics: Antecedents and directions, *Autonomous Robots*, *4*(1), 1997, 7–27.

[13] D. Dudek, M. Jenkin, E. Milios, & D. Wilkes, A taxonomy for multi-agent robotics, *Autonomous Robots*, *3*(4), 1996, 375–397.

[14] P. Stone & M. Veloso, Multiagent systems: A survey from a machine learning perspective, *Autonomous Robotics*, *8*(3), 2000, 345–383.

[15] J. Kim & P. Vadakkepat, Multi-agent systems: A survey from the robot-soccer perspective, *International Journal of Intelligent Automation and Soft Computing*, *6*(1), 2000, 3–17.

[16] T. Balch & R.C. Arkin, Communication in reactive multiagent robotic systems, *Autonomous Robots*, *1*, 1994, 1–25.

[17] K.S. Kwok, B.J. Driessen, C.A. Phillips, & C.A. Tovey, Analyzing the multiple-target-multiple-agent scenario using optimal assignment algorithms, *Journal of Intelligent and Robotic Systems*, *35*, 2002, 111–122.

[18] J. Wein & S.A. Zenios, Massively parallel auction algorithms for the assignment problem, *Proc. 3rd Symp. on Frontiers of Massively Parallel Computation*, College Park, MD, November 1990, 90–99.

[19] P.C. Chu & J.E. Beasley, A genetic algorithm for the generalised assignment problem, *Computers and Operations Research*, *24*(1), 1997, 17–23.

[20] R. Akkiraju, P. Keskinocak, S. Murthy, & F. Wu, An agent-based approach for scheduling multiple machines, *Applied Intelligence*, *14*, 2001, 135–144.

[21] T. Kohonen, Self-organizing formation of topologically correct feature maps, *Biological Cybernetics*, *43*, 1982, 59–69.

[22] C.H.L.K.T. Song, Flexible real-time control of home robots using a multi-agent based approach, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, Sendal, Japan, August 2004, 3092–3097.

[23] J. Wen, H. Xing, X. Luo, & J. Yan, Multi-agent based distributed control system for an intelligent robot, *IEEE Int. Conf. on Services Computing*, Shanghai, China, September 2004, 633–637.

[24] R. Glasius, A. Komoda, & S.C.A.M. Gielen, Neural network dynamics for path planning and obstacle avoidance, *Neural Networks*, *8*(1), 1995, 125–133.

[25] G. Oriolo, G. Ulivi, & M. Vendittelli, Real-time map building and navigation for autonomous robots in unknown environments, *IEEE Trans. Systems, Man, and Cybernetics, Part B*, *28*(3), 1998, 316–333.

[26] S.X. Yang & M. Meng, Neural network approaches to dynamic collision-free trajectory generation, *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, *31*(3), 2001, 302–318.

Biographies

Anmin Zhu received his B.S. and M.S. degree from Tongji University, Shanghai, China, in 1987 and 1990, respectively; received his Ph.D. from the School of Engineering at the University of Guelph, Canada in November 2005. Currently he is a faculty member of the College of Information Engineering, Shenzhen University, China. His research interests include fuzzy systems, neural networks, robotics, self-organizing arrangement, machine intelligence, and multi-agent system.

Simon X. Yang received his B.Sc. degree in engineering physics from Peking University, Beijing, China, in 1987, the first of two M.Sc. degrees in biophysics from Chinese Academy of Sciences, Beijing, China, in 1990, the second M.Sc. degree in electrical engineering from the University of Houston, Houston, USA, in 1996, and his Ph.D. in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 1999.

Dr. Yang joined the University of Guelph, Guelph, ON, Canada as an Assistant Professor in Systems and Computer Engineering in August 1999. Currently he is an Associate Professor and the Director of Advanced Robotics and Intelligent Systems (ARIS) Laboratory at the University of Guelph. His research interests are in the areas of robotics, intelligent systems, sensors, control systems, image and signal processing, neurocomputation, and bioinformatics. He has published over 200 refereed journal papers, book chapters and conference papers.

Dr. Yang serves on the Editorial Boards of the *Dynamics of Continuous, Discrete and Impulse Systems* journal, the *International Journal of Information Acquisition*, and the *International Journal of Computational Intelligence and Applications*. He has involved in the organization of many international conferences. He is the General Chair of the 2006 International Conference on Sensing, Computing and Automation (ICSCA'2006). He was a recipient of the 2004–2006 Presidential Distinguished Professor Awards at the University of Guelph, Canada.