Hybrid autonomous control for heterogeneous multi-agent system

Kazuyuki Ito
Okayama University

Akio Gofuku
Okayama University

Proceedings of the 2003 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Las Vegas, Nevada · October 2003

Hybrid autonomous control for heterogeneous multi-agent system

-Combining of centralized reinforcement learning and distributed rule-based control-

Kazuyuki Ito
Department of Systems Engineering
Okayama University
3-1-1, Tushima-naka, Okayama,Japan
kazuyukiito@ieee.org
www.usm.sys.okayama-u.ac.jp/ kazuyuki/

Akio Gofuku
Department of Systems Engineering
Okayama University
3-1-1, Tushima-naka, Okayama, Japan
fukuchan@sys.okayama-u.ac.jp

Abstract— **Reinforcement learning is an adaptive and flexible control method for autonomous system. In our previous works, we had proposed a reinforcement learning algorithm for redundant systems: "Q-learning with dynamic structuring of exploration space based on GA (QDSEGA)", and applied it to multi-agent systems. However previous works of the QDSEGA have been restricted to homogeneous agents.**

In this paper, we extend our previous works of multi-agent systems, and propose a hybrid autonomous control method for heterogeneous multi-agent systems. To demonstrate the effectiveness of the proposed method, simulations of transportation task by 10 heterogeneous mobile robots have been carried out. As a result effective behaviors have been obtained.

I. Introduction

Reinforcement learning [1] can be an adaptive and flexible control method for autonomous system. It does not need priori knowledge; behaviors to accomplish given tasks are obtained automatically by repeating trial and error. However, as increasing complexity of the system, the learning costs are increased exponentially. So application to the complex systems, like a many redundant degrees of freedom robot and multi-agent system, is very difficult. In the previous works in this field, applications were restricted to simple robots and small size multi-agent systems, and because of restricted functions of the simple systems that have less redundancy, effectiveness of reinforcement learning is restricted, and as a result the abilities (autonomy, adaptability, flexibility) of the previous autonomous system were also restricted.

In our previous works, we had taken these problems into consideration, and had proposed new reinforcement learning algorithm, "Q-learning with dynamic structuring of exploration space based on GA (QDSEGA)[2], [3]".

The QDSEGA is designed for redundant systems and has two class layered structures with one upper agent and many lower agents. The upper agent is a centralized controller of the lower agents and lower agents are distributed controller of real hardware. The upper agent learns how to control the lower agents using reinforcement learning, and lower agents control real hardware to realize desired states that are given by upper agent.

Effectiveness of QDSEGA for redundant systems had been demonstrated using a 12-legged robot[3], [4], a 50-link manipulator[2] and 10 homogeneous mobile robots[5].

However previous works of QDSEGA are restricted to homogeneous agents, and the effectiveness for heterogeneous agents has not been considered.

In the real world, effective systems usually consist of many heterogeneous agents. So the controller should be applicable to various heterogeneous agents, and the controller should assign suitable role to each agent, based on the differences of the agents.

In this paper, we extend our previous works of multi-agent system[5], and propose hybrid autonomous control method for heterogeneous multi-agent system. To demonstrate the effectiveness of the proposed method, simulations of transportation task by 10 heterogeneous mobile robots are carried out.

II. Problem Domain

A. Abilities to realize adaptive autonomous system

We consider an adaptive autonomous control method for heterogeneous multi-agent system based on reinforcement learning. We define key abilities for the adaptive autonomous control as follows.

Autonomy: ability to realize given task without priori knowledge of task and environment.
Flexibility: ability to realize various tasks.
Adaptability: ability to adopt changes of environment and failure.
Applicability: ability that can be applied to heterogeneous large multi-agent system.
Ability to assign: ability to assign suitable role to each

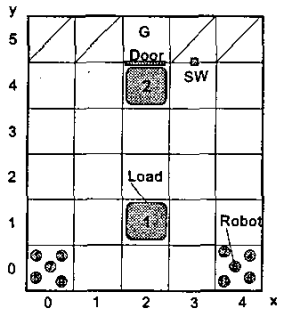

Fig. 1. Transportation Task

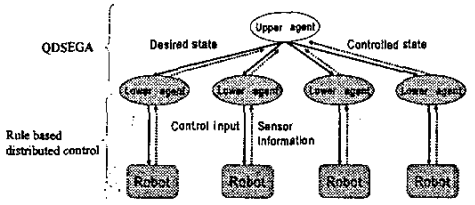

Fig. 2. Layered Structure of Learning architecture

agent.

B. Transportation Task

We consider a transportation task as a typical example. Fig. 1 shows an example of the transportation task. The world consists of 5 × 6 grids. There are 2 loads and 10 mobile robots. G of Fig. 1 means the goal position, and there is a door in front of the goal. SW means a switch to open the door.

An aim of this task is to transport Load 1 to goal position G. The loads have each movable direction. Load 1 can move only vertical direction and Load 2 can move only horizontal direction. The loads are big enough, so only one load can enter into one grid, and any robot cannot enter the grid that is occupied by the load. However the robots are small enough so all robots can share the same grid. To move the load, more than one robots that have enough force must push it toward the same movable direction. Therefore, to realize the task, robots should move the obstacle (Load 2), open the door and move Load 1 to the goal in cooperation.

To realize the task, we employ heterogeneous agents. In this task, the term "heterogeneous" means that each agent has different ability. For example, the force to push the load of each agent is different, some agent can open the door and the other can not.

Next we consider the case that conventional approaches are applied to the task. In case that we use conventional preprogrammed rule-based approaches, the designer must

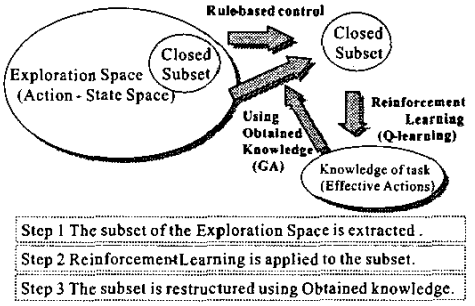

Fig. 3. Outline of Learning Algorithm

assign each role (move obstacle, open door, move load, and so on) to each robot and must implement each rule for each role. So the system can not be autonomous system. In case that we use conventional learning-based approaches, the size of exploration space is expressed as the exponential function of the number of robots. And it is impossible to accomplish the learning because 10 of number of the robots are big enough. Therefore, it is very difficult to realize the task using conventional autonomous control method. So the task is valid to demonstrate the effectiveness and originality of our proposed algorithm.

III. Proposed Method

A. Outline

To realize the adaptive autonomous system, we propose a hybrid control method by connecting preprogrammed rule-based control with the QDSEGA.

At first we explain the outline of the architecture. The QDSEGA has a 2-class layered structure as shown in. Fig. 2. The upper agent, which is a centralized controller based on reinforcement learning, assigns suitable roles to each lower agent and integrates the whole behaviors of the systems. Orders of the upper agent are given to the lower agents as a desired state of the mobile robots. Every lower agent is connected to the mobile robots by a one-to-one correspondence. The lower agents control each mobile robot so that it accomplishes the desired state using preprogrammed rule-based distributed controls. In this paper, we consider the heterogeneous agents, so the lower agents have different rules and different abilities.

Next we explain the flow of learning. At first a subset of the exploration space is extracted (Fig 3 STEP1), next a reinforcement learning algorithm is applied to the subset in order to obtain some knowledge of the task and the environment (Fig 3 STEP2), and then the subset is restructured using the obtained knowledge (Fig 3 STEP3). By repeating the process, effective behaviors of the system are acquired. Details are written in subsections below.

Finally, we confirm that the abilities in subsection II-A are realized in the proposed controller. In the proposed

control method, we have employed reinforcement learning. So the knowledge is obtained by trial-and-error, it does not need priori knowledge and it is applicable to various tasks. Therefore the "autonomy" and "flexibility" is realized.

In the reinforcement learning, a policy to accomplish the task is obtained by interacting to the environment, so the controller can adapt to changes of the environment. Therefore the "adaptability" is realized.

In the proposed controller, the reinforcement learning algorithm is applied to the small extracted subset instead of huge whole exploration space. So it is possible to avoid the state explosion problem. And by restructuring the subset, it is possible to search wide aria. Moreover, to restructure the subset, the controller utilizes obtained knowledge, so the search becomes more efficient than trial-and-error only. Therefore the "applicability" is realized.

The roles of lower agents are assigned by the centralized reinforcement learning algorithm of the upper agent. So relations among the agents are taken into consideration and integrated behaviors of the system are obtained automatically. Therefore the "ability to assign" is realized.

In summary, all abilities in subsection II-A are realized in the proposed controller.

B. Extraction of closed subset

At first, we define an interior state and an exterior state of the upper agent as follows.

The interior state is the set of states that the lower agent can control directly. And the exterior state is the complementary set of the interior state. For example, a position of the mobile robot is an interior state and a position of the load is an exterior state.

A desired state from the upper agent to a lower agent can be regarded as an action of reinforcement learning of the upper agent. In case that the lower agents accomplish the action, which means that each interior state converges to the desired state, a set of actions is equivalent to a set of desired interior state. So by restricting usable actions, the upper agent can restrict necessary interior states, and it becomes possible to extract a closed subset from the exploration space. The term "closed" means that all interior states that can be transited by any actions in the subset are surely contained in the subset. By this nature, we can apply reinforcement learning to the small subset instead of the large exploration space. If the lower agents cannot accomplish an action, which means deadlock, a penalty is imposed to upper agent and new trial is started form the initial state. So the learning process is preceded in the restricted exploration space.

C. Learning Process of Upper Agent

Learning process of upper agent has two dynamics. One is learning dynamics based on Q-learning and the other is structural dynamics based on Genetic Algorithm. Fig. 4 shows the flowchart of the learning algorithm of the upper agent. Each individual of GA expresses an action of Q-learning. At first, an initial set of population is structured randomly, and a Q-table is created using the phenotypes of the individuals in the initial population. The Q-table is reinforced using learning dynamics and the finesses of genes are calculated based on the reinforced Q-table. Selection and reproduction are applied and new population is structured. Repeating this cycle, effective behaviors are acquired. Details are written in subsections below.

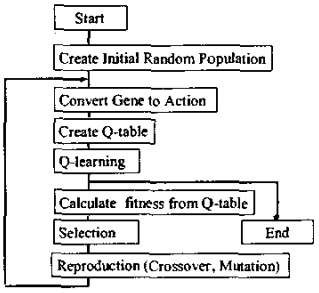

Fig. 4. Flowchart

1) Encode: In this algorithm, each individual expresses the selectable action on the learning dynamics. The number of individuals means the size of the subset of exploration space.

2) Create Q-table: To reduce the redundancy of the actions, the individuals that have a same phenotype are regarded as one action and the Q-table consists of all different actions. The internal states consist of states that can be transited by the generated actions. By repeating the structural dynamics using GA, actions that have a same phenotype are increased, and then the size of the Q-table is decreased.

3) Learning Dynamics: In this paper, the conventional Q-learning [6] is employed as learning dynamics. The dynamics of Q-learning are written as follows.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\{r(s,a) + \gamma \max_{a'} Q(s',a')\} \quad (1)$$

where s is the state, a is the action, r is the reward, α is the learning rate and γ is the discount rate.

4) Fitness of Q-table: The fitness of individuals is calculated by two steps. The first step is regulation of the Q-table and the second step is calculation of the fitness based on the regulated Q-table. At first, we calculate the maximum and the minimum value of the state as follows.

$$V_{max}(s) = \max_{a'}(Q(s,a')), \quad V_{min}(s) = \min_{a'}(Q(s,a'))$$

Then Q' of the regulated Q-table is given as follows

$$\text{if} \quad Q(s,a) \geq 0; \quad \text{then} \quad Q'(s,a) = \frac{1-p}{V_{max}(s)} Q(s,a) + p \quad (2)$$

else $$Q'(s,a) = -\frac{p}{V_{min}(s)}Q(s,a) + p \quad (3)$$

where p is a constant value which means the ratio of reward to penalty. Next, we fix the action a_i and sort $Q'(s,a_i)$ according to their value from high to low for all states, and we define them as the $Q'_s(s,a_i)$ and the operation is repeated for all actions. For example $Q'_s(1,a_i)$ means the maximum value of $Q'(s,a_i)$ and $Q'_s(N_s,a_i)$ means the minimum value of $Q'(s,a_i)$, where N_s is the size of state space. In the second step, we calculate the fitness. The fitness of the individual, whose phenotype is a_i, is given as follows

$$fit_Q(a_i) = \sum_{j=1}^{N_s}\left(w_j\frac{\sum_{k=1}^{j}Q'_s(k,a_i)}{j}\right) \quad (4)$$

where w_i is a weight coefficient which decides the ratio of special actions to general actions.

The fitness defined in (4) has the three important points. The first point is the regularization of the state value of the Q-table. In the Q-learning, as a state becomes closer to the goal, the value of the state becomes higher. So if the fitness is calculated from unregulated Q-table, selected actions at the state that is close to the goal are evaluated as high value. And the actions that are selected near the start state are evaluated as low value, and they are extinguished. However, a series of actions is important to accomplish the task. So the regularization of state value of the Q-table is necessary.

Second point is the handling of the penalty. At the Q-learning, the penalty that has negative value is employed. But the fitness of Genetic Algorithm should be positive, so the conversion of penalty to the fitness is necessary.

Third point is the method of calculation of the fitness. The first term of the equation (4) means the maximum value of the action. When w_1 is chosen as a large value, the action that is effective in special state is evaluated as a high credit, and the special actions are generated by Genetic Algorithm. The last term of (4) implies the mean value of the actions. When w_{N_s} is chosen as a large value, the action that is effective in the various states is evaluated as a high credit, and general actions are generated. Selecting the weight coefficients $(w_1,\cdots,w_{N_s})$, we can set the ratio of the special actions to general actions.

5) Fitness of Frequency of Use: We introduce the fitness of frequency of use to save efficient series of actions. We define the fitness of frequency of use as follows

$$fit_u(a_i) = \frac{N_u(a_i)}{\sum_{j=1}^{N_a}N_u(a_j)} \quad (5)$$

where N_a is a total number of actions at a generation and $N_u(a_i)$ is the number of times which a_i was used in the Q-learning at the generation.

In the fitness of Q-table, the value of series of actions from start to goal is not considered. But to accomplish the task, the series of actions is important and preservation of series is needed.

6) Fitness: Combining (4) and (5) we define the fitness as follows

$$fit(a_i) = fit_Q(a_i) + k_f \cdot fit_u(a_i) \quad (6)$$

where $k_f(k_f \geq 0)$ is a constant value to determine the rate of fit_Q and fit_u.

7) Selection and Reproduction: Various methods of selection and reproduction that have been studied can be applied to our proposed method. The suitable method of the selection and reproduction should be chosen for each given task. In this paper the method of the selection and reproduction is not main subject so the conventional simple GA is employed.

D. Rule-based Control for Lower Agents

In this paper, we employ four basic rules.

R1: The rule to decide a path to a desired position.
R2: The rule to avoid a collision.
R3: The rule to move a load.
R4: The rule to open a door.

Using the rules, each lower agent can make a decision by itself using own local information, and the robots move to the desired positions with pushes the loads and open the door, except for deadlock cases. The deadlock is avoided by planning of upper agent. So the rules do not have to be designed to avoid deadlock. However, if the rules have enough ability to avoid deadlock, the learning of upper agent becomes easer.

Details of implementation of the rules is written in section IV.

IV. SIMULATION

In this section, we demonstrate effectiveness of our proposed method by applying it to the task described in subsection II-B

A. Robots and environment

We consider the grid world described in section II-B and employ 10 robots that have different abilities as written in Table 1. The 2nd column means the force of the robots. The robots have different forces. To move the loads, 2 value of force is needed. So more than one robot must push it. The third column in Table 1 means the type of R1. If there are "normal", the robot employs the R1 written above. If there are " exchange x for y", the robot employs the revised R1 in which x is exchanged for y. The fourth column in Table 1 means the type of R4. If there are "employ", the robot can open the door, and if there are "not employ", the robot does not have ability to open the door.

The controller must assign each role of robot in order to realize the task.

TABLE I

ABILITIES OF ROBOTS

	force	R1	R4
Robot 1	1.0	normal	not employ
Robot 2	0.7	normal	employ
Robot 3	0.7	exchange x for y	employ
Robot 4	1.3	exchange x for y	employ
Robot 5	0.7	normal	not employ
Robot 6	1.3	normal	employ
Robot 7	1.0	exchange x for y	not employ
Robot 8	0.7	exchange x for y	employ
Robot 9	1.0	normal	not employ
Robot 10	0.7	normal	employ

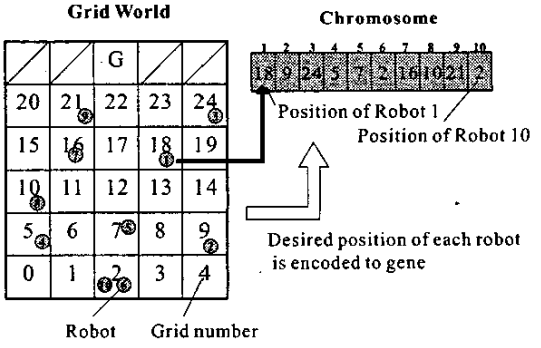

Fig. 5. How to Encode

B. Formation of Proposed Learning Architecture

1) Formation of Rules: We employ rules written as follows.

<**R1**: The rule to decide a path to a desired position>

If $x(i) \neq x_d$ **then**

$x(i+1) = x(i) - \mathrm{sgn}(x(i) - x_d)\Delta x, \quad y(i+1) = y(i)$

Else if $y(i) \neq y_d$ **then**

$y(i+1) = y(i) - \mathrm{sgn}(y(i) - y_d)\Delta y, \quad x(i+1) = x(i)$

Else $x(i+1) = x(i), \quad y(i+1) = y(i)$

Where x_d, y_d: desired position of a robot, $x(i), y(i)$: position of a robot in time i. Using the rule, positions of the robots are calculated step by step.

<**R2**: The rule to avoid a collision>

If *obstacle is on the course that is given by R1* **Then**

If *the obstacle is load* **Then** *Employ R3*

Else *Don't move*

Else *Move using R1*

The collision is divided into 2 types. One is collision between robots and the other is collision to obstacle. To avoid the collision of robots, Traffic Rules were proposed [7] and the effectiveness was demonstrated. So in this simulation, we consider that the grid is enough wide and collision can be avoided using Traffic Rules and we assume that each robot can move to desired direction avoiding other robots.

<**R3**: The rule to move the load>

If *Load is on the course that is given by R1* **Then**

Push the Load to the way that the robot has to go

Else *Move using R1*

<**R4**: The rule to open the door>

If *Switch is in a grid that the robot stopped* **Then**

Turn on the switch to open the door

Else *Nothing is done*

2) Formation of GA: The dynamics of Genetic Algorithm in the proposed algorithm is composed as follows.

We define an action as desired positions of all robots at a step. We number the grid as shown in Fig. 5, and the desired position of a robot is expressed as a grid number. The action is encoded as the genes as shown in Fig. 5. An individual expresses desired positions of all robots at a step. The number of individuals is 300. The roulette selection is employed. The probability of the crossover is 0.2 and uniform crossover is employed. The probability of mutation is 0.001, and 100 times reproduction is carried out. Other parameters are set as follows.

$w_1 = 0.5, w_{Ns} = 0.5, w_i = 0 (i = 2, \cdots, N_s - 1)$,

$k_f = 200$

3) Fomation of Q-learning: The action space consists of the phenotypes of generated individuals. The state space consists of interior state and exterior state. The interior state is composed of the initial state and the states that can be transited by generated actions. The exterior state consists of positions of the loads. Reward is given as follows. When Load1 reaches the goal, 100 is given as a reward, and when Load1 move up or Load2 is removed from the course of Load1, 20 is given as a local reward. A penalty -20 is given when Load1 move down or Load2 block the course of Load1. If the lower agents cannot accomplish an action, a penalty -10 is given and then new trial is started form the initial state.

To select an action, we employ ε-greedy policies. With 10% probability, an action is selected at random. The learning rate is 0.5 and discount rate is 0.9. The number of trials of each learning dynamics is 20000.

C. Simulation Results

Fig. 6 show a typical result. We can find that the task has been completed.

At first we discuss the assigned roles to move the loads. To move Load1, Robot 6 and Robot 9 have been employed. In this simulation, force of Robot 6 is 1.3 and that of Robot 9 is 1.0. So the sum of forces is bigger than 2 which is necessary value to move the load, and both robots have "normal R1", so they can run on the same vertical way.

In case of Load2, Robot 4 and Robot 8 have been employed. The sum of forces is enough too. And both robots have "revised R1", so they can run on the same horizontal way.

We can conclude that necessary cooperative behaviors

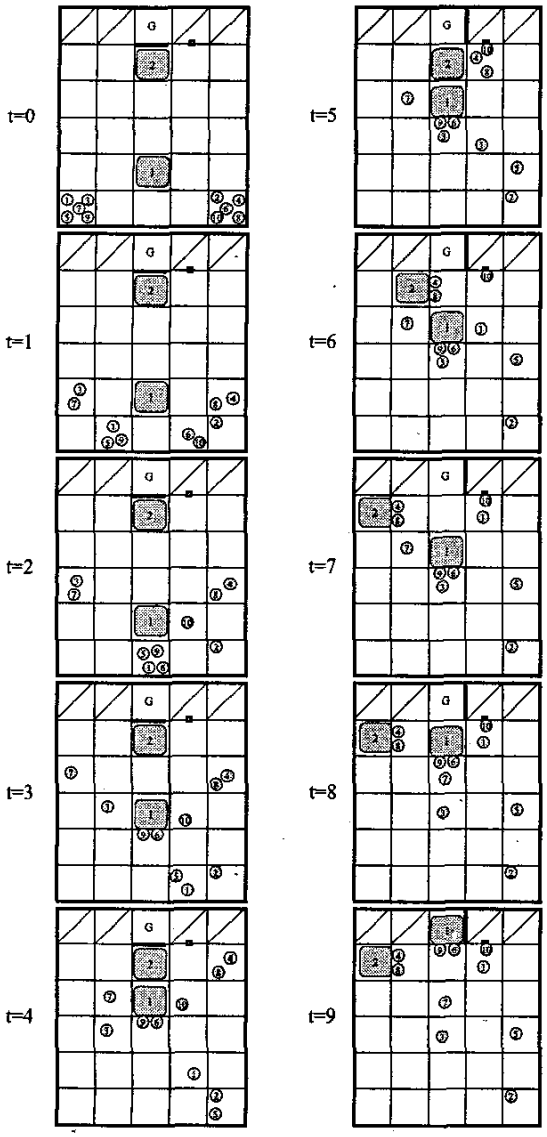

Fig. 6. Simulation result

to move the loads have been assigned automatically and the assignment is valid and effective.

Next we discuss the assigned roles to open the door. The role has been assigned to the Robot 10. In this simulation, Robot 10 has ability to open the door. So we can conclude that the assignment is also valid.

Finally, we consider whole behaviors of the system. In addition to the local assignment of each role, plan of behavior of whole system is necessary to complete the task. From Fig. 6, we can find that the roles of agents have been integrated, and the task has been completed autonomously.

By summarizing the result, we can find that our proposed algorithm has "autonomy", "applicability", and "ability to assign". And in our previous works, "flexibility" and "adaptability" of QDSEGA had been demonstrated[2], [3]. So we can conclude that our proposed algorithm has the abilities that are described in subsectionII-A.

V. Conclusion

We have considered adaptive autonomous control for heterogeneous multi-agent system. To realize the control method, we have extended the QDSEGA and proposed new hybrid control algorithm.

To demonstrate the effectiveness of the proposed algorithm, transportation task has been imposed to the heterogeneous multi-agent system. As a result, suitable roles have been assigned to each different agent, and effective cooperative behaviors have been obtained automatically.

We can conclude that our proposed method is effective for adaptive heterogeneous autonomous multi-agent system.

VI. References

[1] R. S. Sutton. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[2] K. Ito and F. Matsuno. A study of Q-learning: Dynamic structuring of exploration space based on genetic algorithm. *Transactions of the Japanese Society for Artificial Intelligence*, 16(6):510–520(in Japanese), 2001.

[3] K. Ito and F. Matsuno. A study of reinforcement learning for the robot with many degrees of freedom -acquisition of locomotion patterns for multi legged robot-. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 3392–3397, 2002.

[4] K. Ito and F. Matsuno. Applying qdsega to the multi-legged robot. *Transactions of the Japanese Society for Artificial Intelligence*, 17(4):363–372(in Japanese), 2002.

[5] K. Ito, A. Gofuku, and M. Takeshita. A study of reinforcement learning for redundant systems -extend qdsega for multi-agent system-. In *Proc. of the 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, pages 25–32, 2002.

[6] C. J. C. H. Watkins and P. Dayan. Technical note Q-learning. *Machine Learning*, 8:279–292, 1992.

[7] S. Kato, S. Nishiyama, and J. Takeno. Coordinatinng mobile robots by applying traffic rules. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1535–1541, 1992.