Emergence of adaptive behaviors by redundant robots : Robustness to changes environment and failures

Kazuyuki Ito
Okayama University

Akio Gofuku
Okayama University

Emergence of adaptive behaviors by redundant robots -Robustness to changes environment and failures-

Kazuyuki Ito
Okayama University
3-1-1 Tsushima-naka
Okayama, Japan
kazuyukiito@ieee.org

Akio Gofuku
Okayama University
3-1-1 Tsushima-naka
Okayama, Japan
fukuchan@sys.okayama-u.ac.jp

Abstract- Acquiring adaptive behaviors of robots automatically is one of the most interesting topics of the evolutionary systems. In my previous works, we have developed an adaptive autonomous control method for redundant robots. The QDSEGA is one of the methods that we have proposed for them. The QDSEGA is realized by combining Q-learning and GA, and it can acquire suitable behaviors by adapting a movement of a robot for a task. In this paper, we focus on the adaptability of the QDSEGA and discuss the robustness of the autonomous redundant robot that is controlled by the QDSEGA. To demonstrate the effectiveness of the QDSEGA, simulations of obstacle avoidance by a 10-link manipulator in the changeable environment and locomotion by a 12-legged robot with failures have been carried out, and as a result, adaptive behaviors for each environment and each broken body have emerged.

1 Introduction

Acquiring adaptive behaviors of robots automatically is one of the most interesting topics of the evolutionary systems. In that field, redundant robots, that have many degrees of freedom, are one of interesting and difficult application. The many degrees of freedom of the robot make it possible to realize various behaviors, and the redundant robots have high adaptability to changes of environment and failures of the body.

In the conventional works, reinforcement learning [Sutton 98] has applied to robots to control them autonomously[Kimura 99, Doya 01]. By the reinforcement learning, robots can learn effective behaviors autonomously without priori knowledge by repeating trial and error.

However in applying the reinforcement learning to the robot with many degrees of freedom, there was significant problem. The many degrees of freedom cause state explosion problem and it makes it difficult to learn.

In our previous works, we have considered how to treat the state explosion problem of the redundant robots, and we have developed an autonomous control methods for them[Ito 01a, Ito 01b, Ito 02a]. The "Q-learning with Dynamic Structuring of Exploration Space Based on Genetic Algorithm[Ito 01b, Ito 02b]" is one of them. It is realized by combining Q-learning and GA, and it can acquire suitable behaviors by adapting a movement of a robot for a task. The effectiveness of the QDSEGA has been demonstrated using 50-link manipulator[Ito 01b], 12-legged robot[Ito 02b], 10 mobile robots[Ito 02a, Ito 03a] and real snake-like robot[Ito 03b], and flexibility to the differences of bodies and differences of tasks have been discussed.

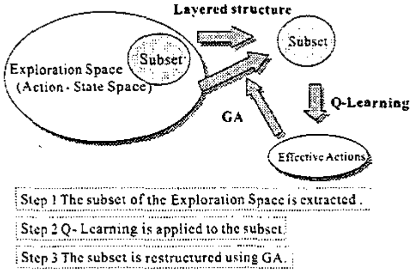
Layered structure
Exploration Space
(Action - State Space)
Subset
Subset
Q-Learning
GA
Effective Actions
Step 1 The subset of the Exploration Space is extracted .
Step 2 Q- Learning is applied to the subset.
Step 3 The subset is restructured using GA.

Figure 1: Outline of QDSEGA

However, adaptability to changes of environment and failures of the robots has not been discussed. The robustness against the changes of environment and the failures of the robots is also very important ability for autonomous robots.

In this paper, we focus on the adaptability of the QDSEGA and discuss the robustness of the autonomous redundant robot that is controlled by the QDSEGA. To demonstrate the effectiveness of the QDSEGA, simulations of obstacle avoidance by a 10-link manipulator in the changeable environment and locomotion by a 12-legged robot with failures are carried out.

2 QDSEGA

2.1 Outline

Fig. 1 shows the outline of QDSEGA. The learning process is as follows. At first, small subset of exploration space is extracted from the large exploration space which is composed of state space and action space. Next, reinforcement learning is applied to the subset and some knowledge of the task is obtained. And then new subset of the exploration space is created using the acquired knowledge. The reinforcement learning

0-7803-7804-0 /03/$17.00 © 2003 IEEE

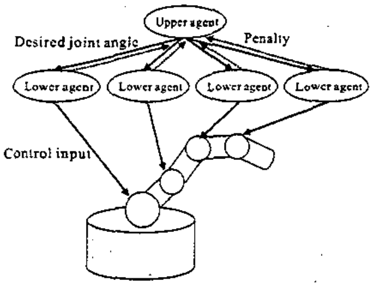

Figure 2: Layered Structure

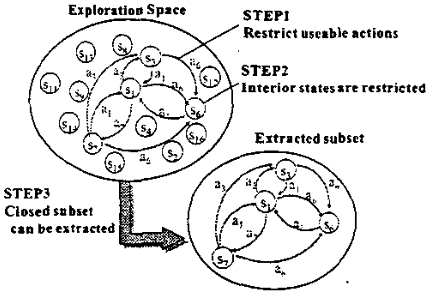

Figure 3: Extraction of Closed-subset

is applied to the new subset. By repeating this cycle, effective subset and effective policy in the subset is acquired.

By extracting the closed-subset, it becomes possible to apply the reinforcement learning to the small extracted exploration space. And by using acquired knowledge to restructure the subset, the search becomes more efficient compare to trial and error only.

The function to extract the subset is realized by layered structure of learning architecture and the reinforcement learning is realized by Q-learning. The subset is restructured using genetic algorithm.

2.2 Interior State and Exterior State

In this paper, we define an interior state and an exterior state as follows. The interior state is the set of states that the agent can control directly. And the exterior state is all the states other than the interior states. For example, in considering obstacle avoidance by a manipulator, every joint angle of the manipulator is interior state and positions of the obstacle are exterior state.

2.3 Layered Structure

Proposed algorithm has 2 level layered structures. Fig. 2 shows an example of application to a manipulator. The upper agent plans all trajectories of interior state, and passes them to the lower agents as desired states. Each lower agent corresponds to an actuator of the robot by one to one, and controls each joint angle so that it becomes the desired state.

The communications between the upper agent and the lower agents is two-way, and if a lower agent can not realize the desired state that is given by the upper agent, the lower agent returns the information to the upper agent as a penalty. If the upper agent catches a penalty from any lower agent, the upper agent withdraws the desired states that is given to lower agents, and plans new desired states. So by repeating a learning process, desired states that can not be realized by lower agents are rejected, and a trajectory that complete given task is composed of only realizable states.

By the layered structure, the proposed algorithm can be applied to the real systems that have dynamics and complexity, because the lower agents are interfaces of real hardware system to software systems.

2.4 Extraction of Closed Subset

A set of desired states that are given by the upper agent to the lower agents at a step 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 (a_1, a_3, a_6, a_7 in Fig. 3 STEP1), the upper agent can restrict necessary interior states (s_1, s_3, s_6, s_7 in Fig. 3 STEP2), and it becomes possible to extract a closed subset from the exploration space(Fig. 3 STEP3). The term "closed" means that the interior state that can be changed by an action in the subset is surely contained in the subset. By this way, we can apply reinforcement learning to the small subset instead of the large exploration space.

If the lower agents cannot accomplish an action, a penalty is imposed to upper agent and new trial is started form the initial state. So the learning process can be preceded in the restricted exploration space.

We can structure the subset of exploration space dynamically by structuring the action space dynamically. In the proposed algorithm, the actions are selected using genetic algorithm in the learning process of the upper agent. Details are written in subsection 2.5.

2.5 Learning Process of Upper Agent

The proposed algorithm has two dynamics. One is a learning dynamics based on Q-learning and the other is a structural dynamics based on Genetic Algorithm. Fig. 4 shows the flowchart of the learning process of the upper agent.

Each action is expressed as a phenotype of genes and restructured by Genetic Algorithm. At first, an initial set of population is structured randomly, and the Q-table that consists of phenotype of the initial population is constructed. 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

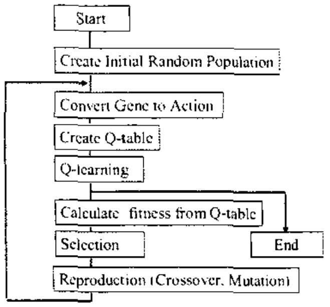
Start
Create Initial Random Population
Convert Gene to Action
Create Q-table
Q-learning
Calculate fitness from Q-table
Selection
End
Reproduction (Crossover, Mutation)

Figure 4: Learning process of the upper agent

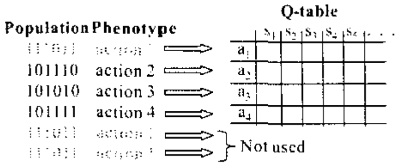

Figure 5: Create Q-table

structured. Repeating this cycle, effective behaviors are acquired. Details are written in subsection 2.6–2.9.

2.6 Encoding

In this algorithm, actions of the Q-learning are encoded to chromosome. Each individual expresses the selectable action on the Q-learning as written in Fig. 5. It means that subsets of actions are selected and learning dynamics is applied to the subset. The subset of action is evaluated and a new subset is reconstructed using Genetic Algorithm. The number of individuals means the size of the subset. Details are written in section 3.1.

2.7 Create Q-table

To reduce the redundancy of actions, the genes that have a same phenotype are regarded as one action and the Q-table consists of all different actions as depicted in Fig. 5. The interior 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.

2.8 Learning dynamics

In this paper, the conventional Q-learning[Watkins 92] is employed as a 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.

2.8.1 Fitness of Q-table

The fitness of chromosome is calculated at two steps. The first step is regulation of the Q-table and the second step is calculation of the fitness from the regulated Q-table.

At first, we calculate the maximum and minimum value of the state as follows.

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

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

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

if $Q(s,a) \geq 0$ then

$$Q'(s,a) = \frac{1-p}{V_{max}(s)}Q(s,a) + p \qquad (2)$$

else

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

where p is a constant value which means the ratio of reward to penalty. Next, we fix the action to 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 repeating the operation 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 gene whose phenotype is a_i is given as follows

$$\begin{aligned} fit_Q(a_i) &= w_1\frac{Q'_s(1,a_i)}{1} + w_2\frac{Q'_s(1,a_i)+Q'_s(2,a_i)}{2} + \\ &\cdots + w_{N_s}\frac{Q'_s(1,a_i)+Q'_s(2,a_i)+\cdots+Q'_s(N_s,a_i)}{N_s} \\ &= \sum_{j=1}^{N_s}\left(w_j\frac{\sum_{k=1}^{j}Q'_s(k,a_i)}{j}\right) \qquad (4) \end{aligned}$$

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

The fitness of Q-table has the three important points. The first point is the regularization of the state value of the Q-table. In the Q-learning, the value of the state that is closer to goal state is higher. So if the fitness is calculated from unregulated Q-table, the selected actions at the state that is close to the goal are evaluated as a high values. And the actions that are selected near the start state are evaluated as a low values and they are extinguished. But to accomplish the task, a series of actions is important. 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 usually should be positive, so the conversion of the penalty to the fitness is necessary. At the proposed method, the positive value of the Q-table is converted to the value from p to 1.0 and the negative value converted to the value from 0 to p. We can choose the rate of the reward to the penalty by selecting the value of p.

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 the 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. And 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, we can set the ratio of the special actions to general actions. And in the proposed algorithm, the special actions and the general actions are evaluated as a high credit simultaneously, so they can coexist. And our proposed algorithm does not have "don't care" symbol that is used in classifire system[Holland 86], so the problems caused by using "don't care" symbol[A. Hayashi 99] do not exist.

2.8.2 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

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

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

In the fitness of the Q-table written in subsection 2.8.1, the values of actions at each state are evaluated independently of the other actions. However, to realize a task, group of actions that move a state from start to goal is necessary. If one action in the group is extinguished, the task can not be realized and every action in the group loses their values. So the fitness that evaluates the group is necessary. In other words, the proposed system is a co-evolutionary system, so the fitness that can treat co-evolution is necessary. By the fitness of frequency of use, we can evaluate the group of actions, because in the learning dynamics (Q-learning), effective behavior is repeated and the group of actions that compose the effective behavior is used frequently.

2.8.3 Fitness

Combining discussion in the sections 2.8.1, 2.8.2 we define the fitness as follows

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

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

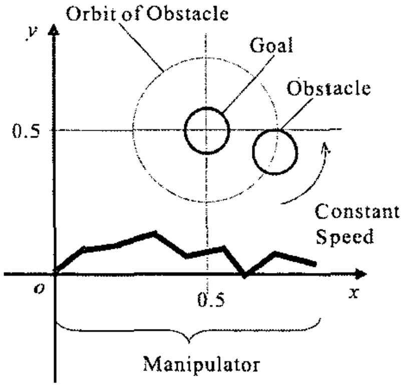

Figure 6: Obstacle avoidance using redundant manipulator

2.9 Selection and Reproduction

Various methods of selection and reproduction that have been studied can be applied to our proposed algorithm. The 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 we embed our algorithm on the basis of simple GA. Details are written in section 3.1.

3 Adaptively for the changes of environment and failure

In this section, we consider the adaptability of the proposed method in applying it to the robot with many redundant degrees of freedom.

3.1 Adaptability to changes of environment

In this subsection, we consider adaptability to the changing environment of the proposed algorithm. We employ the task of obstacle avoidance of a manipulator, and we carried out two cases. Case 1 is the case that the goal position is changed and Case 2 is the case that the orbit of obstacle is changed.

3.1.1 Task

We apply the proposed method to the problem of obstacle avoidance using a 10-link manipulator. Let us define the origin and coordinate as shown in Figure 12. The origin means the fixed end of manipulator and the first joint angle is the angle from the x-axis. The goal of the task is taking the top of the manipulator to the interior of the desired circle with avoiding the obstacle. The length of manipulator is 1.5 and the initial attitude is the straight line on the x-axis. The obstacle has a circle shape whose radius is 0.1 and its center moves on the circle trajectory whose center is (0.5, 0.5) and radius is 0.3 with constant speed. And the initial position on the circle is random. The goal region is the

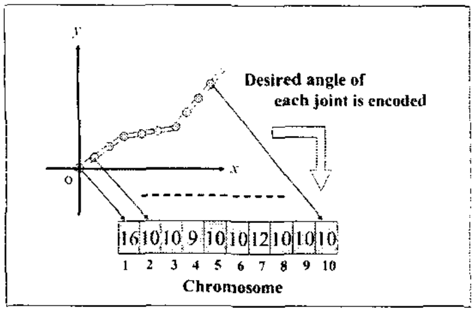

Figure 7: How to encode the actions

0.1 radius circle and the center is (0.5,0.5) and does not move.

3.1.2 Simulation model of the manipulator

We regard that all actuators are stepping motors and the angle and angular velocity can be controlled. We assume that all joints are moved to the desired angle by the same constant angular velocity. And when the joint reaches the desired angle, the joint is stopped. And when all joint angles reach the desired angles, the manipulator is stopped.

3.1.3 Simulation

<Formation of genetic algorithm> At first we describe the encoding. We define an action as desired angles of all joints at a step. And one action is encoded as one chromosome as shown in Fig. 7. The chromosome has a same number of genes as the number of joints. One gene expresses an angle of one joint. One gene has 9 characters that express the angles from -40[deg] to 40[deg] every 10 degrees. The numbers of individuals is 200. And roulette selection is employed. The probability of the crossover is 0.5 and one-point crossover is employed. The probability of mutation is 0.004. The chromosome whose phenotype shows the cross shape of the manipulator is regarded as lethal genes and is extinguished.

<Formation of Q-learning>

The exterior states consist of the positions of obstacle. The movable region of the obstacle is divided into 20 parts. When the top of manipulator reaches the goal region, the value of 100 is given as a reward. When any part of the manipulator touches the obstacle, the value of -50 is given as a penalty. The roulette selection using Boltzmann distribution is employed. The learning rate is 0.5 and discounting rate is 0.9. The number of trials of each learning dynamics is 1000 times.

<Simulation>

We carried out two cases. Case 1 is the case that the goal position is changed as written in Fig. 8, and Case 2 is the case that the orbit of obstacle is changed as written in Fig. 9. At first, the learning process is carried out in the normal environment that is shown in Fig. 6. After 30 reproductions, the environment is changed, and another 20 reproductions are carried out. Two different simulations are carried out for each case respectively.

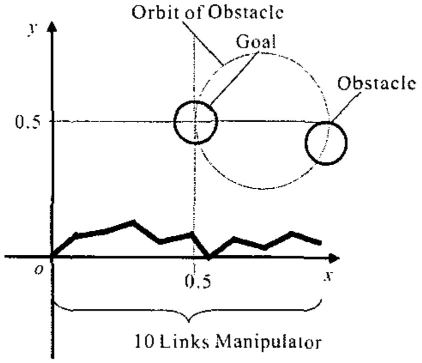

Figure 8: Change of goal position

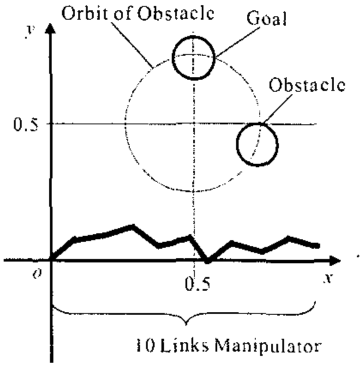

Figure 9: Change of obstacle position

<Results>

Fig. 10 shows an acquired behavior at the normal condition. We can find that the effective behavior is acquired and the task is accomplished.

Fig. 11 shows the transition of gains. The gain is calculated by dividing the total reward by total step in a generation. From Fig. 11 we can find that in Case 1, the gain of 31th generation is positive though the environment has been changed. It means that the task has been completed in spite of the change of the environment. The proposed algorithm can adapt the behavior to the changed environment, within the learning dynamics (Q-learning). It means that it is possible to adapt the changes by only changing the timing to use the actions. It does not have to change the actions itself. So the task has been completed using only the presented actions.

On the other hand, in the Case2, the gain of 31th generation becomes negative and after a few reproductions, the gain returns to positive. It means that it is impossible to adapt the behavior to the changed environment within the generation, but it can adopts it to

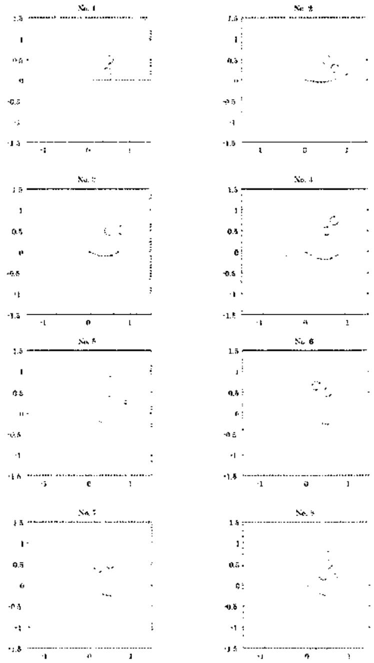

Figure 10: Acquired behavior (Task of obstacle avoidance by 10-link manipulator)

the environment after a few generations. In Case2, the position of the goal is changed, so the task can not be realized by only changing the timing to use the actions. Therefor the actions must be restructured for new goal position. In the proposed algorithm, the actions are restructured automatically to adapt the behavior to the changes of the environment and after a few generations, the task is realized in the new environment.

We can consider that the proposed algorithm has two kind of adaptability. If the changes of environment are small enough, the algorithm can adapt the behavior to the environment quickly by the reinforcement learning, and even if the changes of environment are not small, the algorithm also can adapt the behavior to the environment by restructuring the actions. Moreover the suitable adaptation is selected automatically against the scale of the changes and be executed automatically.

We can conclude that the adaptability of the proposed algorithm is effective and robust against the changes of environment.

3.2 Acquisition of Locomotion Patterns by Multi-legged Robot

In this subsection we consider the robustness of the proposed algorithm for the fault of the robot. We employ the locomotion task of multi-legged robot and we consider the case when some legs of the robot are broken.

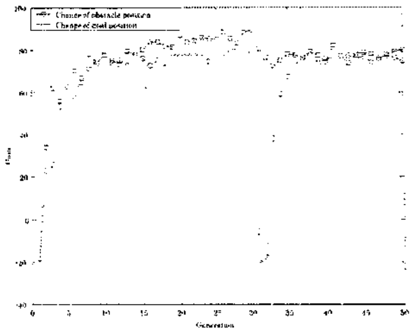

Figure 11: Transition of gains

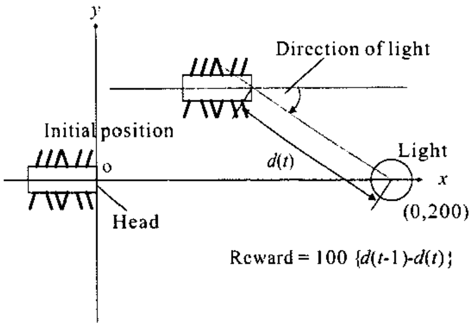

Figure 12: Task of locomotion

3.2.1 Task

The task is how to get closer to the light source without tumble. Fig. 12 shows the outline of the task. The light source is far enough from the start position and the reward is calculated using the distance from the light source.

3.2.2 Simulation Model

We employ Minimal Simulation Model that was proposed by M. Svinin et al., [Svinin 99]. This model is very simple and it can be calculated very low computational cost.

Fig. 13 shows a multi-legged robot. Each leg has two joints and has four motions (Move forward and lift down, Move back and lift down, Move forward and lift up, Move back and lift up). The position of the robot can be calculated as follows.

$$f^r_{drv} = u(n^r_{12} - n^r_{21}) \tag{7}$$

$$f^r_{res} = v(n^r_{11} + n^r_{22}) \tag{8}$$

$$F^r = \begin{cases} 0 & \text{if } |f^r_{drv}| \le f^r_{res} \\ f^r_{drv} - f^r_{res} & \text{else if } f^r_{drv} > f^r_{res} \\ f^r_{drv} + f^r_{res} & \text{else } (f^r_{drv} < -f^r_{res}) \end{cases} \tag{9}$$

$$F = F^l + F^r \tag{10}$$

$$M = F^r - F^l \tag{11}$$

$$\Delta u = c_u F \tag{12}$$

$$\Delta \theta = c_\theta M \tag{13}$$

$$x(t+1) = x(t) + \Delta u \cos\theta(t) \tag{14}$$

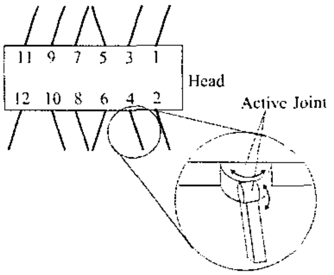

Figure 13: Multi-legged Robot

$$y(t+1) = y(t) + \Delta u \sin\theta(t) \tag{15}$$

$$\theta(t+1) = \theta(t) + \Delta\theta \tag{16}$$

where n_{ij}^r is the number of legs on the right side changing their configuration from the state i at the moment t to the state j at the moment $t+1$. Similarly we define n_{ij}^l for the left side. Let F and M be the force and the moment of the robot that act to the environment, respectively. And x,y,θ mean the position and orientation of the robot. Details are written in [Svinin 99].

3.2.3 Simulation

<Formation of GA>
The dynamics of Genetic Algorithm of the proposed algorithm is composed as follows.

At first we describe the coding. We define the action as the desired state of legs. One action expresses desired states of all legs at a step. One action is encoded as one chromosome. The chromosome has a same number of genes as the number of legs. One gene expresses desired state of one leg. The number of individuals is 200. 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.
<Formation of Q-learning>
The action space consists of the phenotypes of the generated genes. The state space consists of interior states and exterior states. The interior states are composed of the initial state and the states that can be transited by generated actions. The exterior states consist of the angle to the goal. The angle is divided four states (form -45[deg] to 45[deg] each 30[deg]). Reward is calculated as follows and it is given each step.

$$r(t) = 100(d(t-1) - d(t)) - |y(t) - y(t-1)| \tag{17}$$

If the robot tumbles (When center of gravity is not within the polygon that consist of the legs that contact the grand), the penalty -100 is given and the trial is finished and next trial is started from the initial position.

The roulette selection using the Boltzmann distribution is employed.The learning rate is 0.5 and discounting rate is 0.9. The number of trials of each learning dynamics is 10000 times.

At first, the learning process is carried out using the normal robot. After 100 reproductions, the leg 1 and 3 in Fig. 13 is broken down, and another 100 reproductions are carried out.

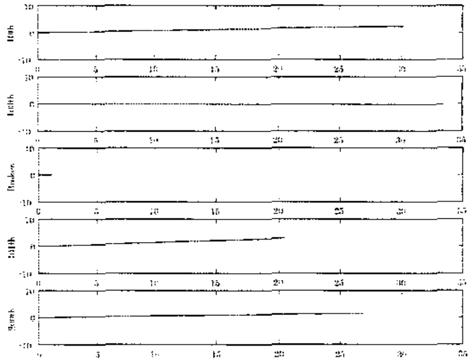

Figure 14: Track of locomotion at each generation

<Results>
Fig . 14 shows tracks of realized locomotion at each generation. The numbers that are written at the left of Fig. 14 means the generations. At 10th generation, the distance that the robot can move is short and the direction is not on right direction. By repeating the reproductions, the locomotion is improved, and at 100th generation, the robot can move to the right direction and the movable distance is almost optimal.

When the legs broken, the robot can not move. However, at 101th, the robot learns to move using broken body. After repeating reproductions, the movable distance becomes longer and at 200th generation, the robot has acquired the optimal behavior in the broken body. It means that the proposed algorithm has two level of adaptability. At first the robot can adapt itself to the failure quickly by the learning dynamics, and then the system improves the behaviors by structural dynamics.

Fig. 15 shows an acquired locomotion pattern of each leg at 200th generation. The number of left side means the number of the leg. The horizontal axis means steps and the vertical axis means condition of the leg (1: Move forward and lift down, 2: Move back and lift down, 3: Move forward and lift up, 4: Move back and lift up). From Fig. 15, we can find that the Leg1 and Leg3 are broken and they do not move and they can not hold the body, and the others legs have certain patterns. In this simulation model, the optimum pattern of each leg that maximizes the moving distance is 1, 2, {3 or 4}, 1, 2, {3 or 4}. So, the Leg4–Leg7 and Lg9–Leg12 have optimal pattern, and the patterns of Leg2 and Leg 8 are not optimal and these legs does not contribute to the movement of the body. When the body was normal, every leg had optimal pattern (details are written in [Ito 02b]). So we can find that the patterns of Leg2 and Leg8 are changed adaptively. In this case, both broken legs are on the left side of the body. So if every leg that are not broken move optimally, the body turn to the left. The robot

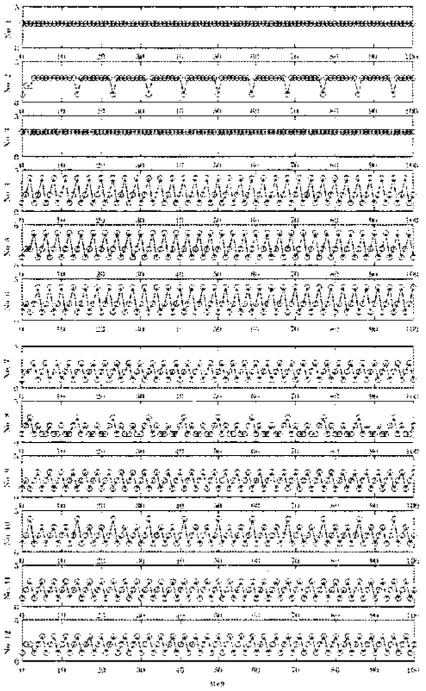

Figure 15: Locomotion pattern

can not move straight. We can consider that the robot reduced the forces of Leg2 and Leg8 in order to balance the forces of both sides. because the Leg2 and Leg8 are on the right side. The Leg2 and Leg8 move only in order to hold the body not to tumble. The robot can move straight with optimal speed at the broken body, without tumble.

We can conclude that the adaptability of the proposed algorithm is effective and robust for the failures.

4 Conclusion

In this paper, we have considered an adaptive and autonomous control method for redundant robots. We have focused on the adaptability of the QDSEGA and discussed the robustness of the autonomous redundant robot that was controlled by the QDSEGA.

To demonstrate the effectiveness of the QDSEGA, simulations of obstacle avoidance by a 10-link manipulator in the changeable environment and locomotion by a 12-legged robot with failures have been carried out, and as a result, adaptive behaviors for each environment and each broken body have emerged.

The adaptation of the QDSEGA is very effective because the system has 2 different dynamics to adapt itself to changes. At first, the system can adapt itself to changes quickly by the Q-learning in the local exploration space and after that the system can explore more suitable behavior from the global exploration space by the GA. The quick adaptation in the local exploration space and slow adaptation in the global exploration space are compatible with each other.

We can conclude that the QDSEGA is effective for adaptive and autonomous control of redundant robots.

Bibliography

[A. Hayashi 99] A. Hayashi. N. S.: GLS: a Hybrid Classifier System Based on POMDP Research. *Transactions of the Japanese Society for Artificial Intelligence*. Vol. 14. No. 3. pp. 538–546(in Japanese) (1999).

[Davis 91] Davis. L.: *Handbook of Genetic Algorithms*. Van Nostrand Reinhold (1991).

[Doya 01] Doya. K.. Kimura. H.. and Kawato. M.: Neural Mechanisms of Learning and Control, *IEEE Control Syatems Magazine*. Vol. 21. No. 4. pp. 42–44 (2001).

[Holland 86] Holland. J. H.: Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based system.. in *Machine Learning II*. pp. 593–623 (1986).

[Ito 01a] Ito. K. and Matsuno. F.: Application of reinforcement learning to hyper-redundant system -Acquisition of locomotion pattern of snake like robot-. in *Proc. The Pacific Asian Conference on Intelligent Systems*. pp. 65–70 (2001).

[Ito 01b] Ito. K. and Matsuno. F.: A Study of Q-learning: Dynamic Structuring of Exploration Space Based on Genetic Algorithm. *Transactions of the Japanese Society for Artificial Intelligence*. Vol. 16. No. 6. pp. 510–520(in Japanese) (2001).

[Ito 02a] Ito. K.. Gofuku. A.. and Takeshita. M.: 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*. pp. 25–32 (2002).

[Ito 02b] Ito. K. and Matsuno. F.: Study of Reinforcement Learning for the Robot with Many Degrees of Freedom -Acquisition of Locomotion Patterns for Multi Legged Robot-. in *Proc. IEEE Int. Conf. on Robotics and Automation*. p. to appear (2002).

[Ito 03a] Ito. K.. Gofuku. A.. and Takeshita. M.: Hybrid autonomous control for heterogeneous multi-agent system -Combining of centralized reinforcement learning and distributed rule-based control-. in *Proc. of 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (2003). to appear.

[Ito 03b] Ito. K.. Kamegawa. T.. and Matsuno. F.: Extended QDSEGA for Controlling Real Robots -Acquisition of Locomotion Patterns for Snake-like Robot-. in *Proc. of IEEE Int. Conf. on Robotics and Automation* (2003). to appear.

[Kimura 99] Kimura. H. and Kobayashi. S.: Efficient Non-Linear Control by Combining Q-learning with Local Linear Controllers. in *Proc. of 16th International Conference on Machine Learning*. pp. 210–219 (1999).

[Sutton 98] Sutton. R. S.: *Reinforcement Learning: An Introduction*. The MIT Press (1998).

[Svinin 99] Svinin. M.. Ushio. S.. Yamada. K.. and Ueda. K.: Emergent systems of motion patterns for locomotion robots. in *Proc. of Int. Workshop on Emergent Synthesis*. pp. 119–126 (1999).

[Watkins 92] Watkins. C. J. C. H. and Dayan. P.: Technical Note Q-Learning. *Machine Learning*. Vol. 8. pp. 279–292 (1992).