Control of hyper-redundant robot using QDSEGA

Kazuyuki Ito
Okayama University

Fumitoshi Matsuno
Tokyo Institute of Technology

SISE02-0460

Control of Hyper-Redundant Robot using QDSEGA

Kazuyuki Ito[1], Fumitoshi Matsuno[2]
1 Okayama University, 3-1-1, Tsushimanaka, Okayama-city, Okayama, Japan
2 Tokyo Institute of Technology, 4259, Nagatuta, Midori, Yokohama, Japan

Abstract: We consider a flexible autonomous system. To realize the system, we employ Hyper-redundant system (It is flexible hardware system) and Reinforcement learning controller "QDSEGA" (It is a flexible software system). In this paper we apply QDSEGA to controlling of Hyper-redundant robot. To demonstrate the effectiveness, a task of acquisition of locomotion patterns is applied to a multi-legged formation and a snake-like formation, as a result effective locomotion has been obtained.

Keyword: Hyper-redundant systems, Q-learning, Reinforcement learning, Genetic Algorithm, QDSEGA

1 Introduction

Recently, flexible autonomous system that can accomplish various tasks automatically has been much attention. However most of conventional researches of autonomous system are restricted to some simple robots like a mobile robot [1][2], so it cannot accomplish various tasks. On the other hand, in the field of researches of redundant robot[3][4][5], the robot can accomplish various tasks using its redundancy but it requires control methods for each tasks. So it cannot be autonomous system. Considering these points, we propose new approach to realize autonomous system. To realize the system, we employ a hyper-redundant robot and a reinforcement learning controller "QDSEGA[6]". The hyper-redundant robot is composed of many uniform units and separation and recombination is possible. Using its redundancy the hyper-redundant robot can accomplish various tasks. On the other hand, QDSEGA (Q-learning with dynamic structuring exploration space based on GA) is a one of the reinforcement learning approach for the robot with many redundant degrees of freedom and it can be flexible autonomous controller. Using QDSEGA, control method of the robot to accomplish a task is acquired automatically by repeating try and error.

We can consider that the hyper-redundant robot is a flexibility of hardware and QDSEGA is a flexibility of software. By combining these adaptive hardware system and software system, real adaptive system can be constructed. So controlling of hyper-redundant robot using QDSEGA is very attractive and it might be effective. In this paper, we apply QDSEGA to the controlling of hyper-redundant robot. To demonstrate the effectiveness, a task of acquisition of locomotion patterns is applied to a multi-legged robot and a snake-like robot.

2 Hyper-redundant system

Fig. 1 shows a unit of a hyper-redundant robot. The unit has one active joint and it can combine other units. A hyper-redundant robot is composed of many uniform units and separation and recombination are possible. By changing the form of combination, the hyper-redundant system can adapt itself to various environment and various given tasks. Fig. 2 - Fig. 4 show the typical forms

Figure 1: Basic unit

Figure 2: Manipulator

Figure 3: Snake-like robot

Figure 4: Multi-legged robot

of the hyper-redundant robot. The multi-legged robot and the snake-like robot are one of the typical and simple moving forms of the hyper-redundant systems. In this paper we consider the multi-legged forms and the snake-like forms as a typical example.

3 QDSEGA

In this section we explain our proposed reinforcement learning algorithm QDSEGA [6].

3.1 Learning architecture

Learning architecture is designed for redundant robots based on three basic ideas. The first one is the to construct a closed set of pairs of actions and states so as to

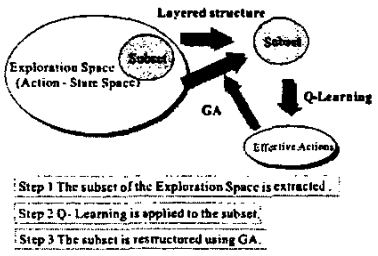

Figure 5: Outline of Learning architecture

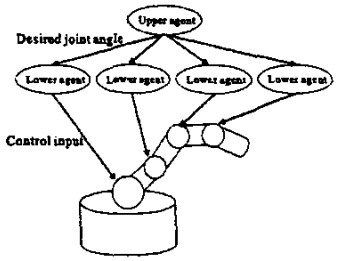

Figure 6: Hierarchal structure

restrict exploration space. The second one is to apply reinforcement learning to the restricted small exploration space so as to acquire some knowledge to accomplish a task. The third one is to restructure restricted small exploration space dynamically using acquired knowledge so as to explore wide area.

3.2 Q-learning with Dynamic Structuring of Exploration Space Based on GA

In consideration of the ideas in the section 3.1, QDSEGA is designed. In this method, the idea of restricting exploration space is realized by hierarchal structure, reinforcement learning is implemented by Q-learning and the exploration space is restructured using genetic algorithm (GA) as depict in Fig. 5.

3.2.1 Interior State and Exterior State

We focus on the body of redundant robot and we classify state space into two categories: state of robot (interior state) and state of environment (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 the complementary set of the interior state.

3.2.2 Hierarchal Structure

Proposed method has a 2-class hierarchal structure. Fig. 6 shows an example of the hierarchal structure for a manipulator. An upper agent plans all trajectories of the interior states, and passes them to lower agents as desired states. A lower agent controls each interior state so that it accomplishes the desired state.

The lower agent can be implemented easily using conventional method for example control theoretic approaches, stepping motors and so on. In this paper, we mainly consider the learning algorithm of upper agent .

3.2.3 Dynamic Structuring of Exploration Space

A desired state from the upper agent to a lower agent can be regarded as an action. If the lower agents accomplish that each interior state converges to the desired state

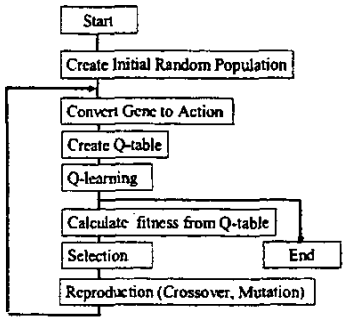

Figure 7: Flowchart

from the upper agent, a set of actions is equivalent to a set of the interior state that can be transited. So restricting the action space can restrict the interior state space and as a result the exploration space can be restricted. And we can structure the exploration space dynamically by structuring the action space dynamically.

If the lower agents cannot accomplish that each interior state converges to the desired state from the upper agent, 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.

3.2.4 Outline of Learning Process of Upper Agent

Learning process of uper agent has two dynamics. One is learning dynamics based on Q-learning and the other is structural dynamics based on Genetic Algorithm. Fig. 7 shows the flowchart of the proposed algorithm 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 structured. Repeating this cycle, effective behaviors are acquired.

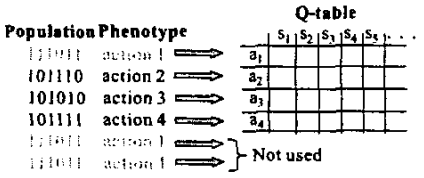

Figure 8: Construction of Q-table

3.2.5 Encode

In this algorithm, each individual expresses the selectable action on the learning dynamics. It means that subset of actions is selected and learning dynamics is applied to the subset. The subset of action is evaluated and a new subset is restructured using Genetic Algorithm. The number of individuals means the size of the subset. If size of subset is too small, it is impossible to accomplish the tasks and Genetic Algorithm does not work effectively. So the size of the population should be chosen large enough. The necessary size is only depending on the complexity of the given task. In our proposed algorithm, the number of actions (that consist of the Q-table) is reduced as the learning process. This is advanced feature. So our proposed algorithm works effectively even when the size of population is large.

3.2.6 Create Q-table

To reduce the redundancy of actions, the genes that have same phenotype are regarded as one action and the Q-table consists of all different actions (Fig. 8). And the size of the Q-table is decreased with the repeating of structural dynamics using Genetic Algorithm.

3.2.7 Learning Dynamics

In this paper, the conventional Q-learning [7] 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.

3.2.8 Fitness of Q-table

The fitness of genes 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 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

$$\text{if} \quad Q(s,a) \geq 0; \quad \text{then}$$

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

$$\text{else} \quad (Q(s,a) < 0)$$

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

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 gene whose phenotype is related to the action a_i is given as follows

$$fit_Q(a_i) = w_1 \frac{Q'_s(1,a_i)}{1} + w_2 \frac{Q'_s(1,a_i) + Q'_s(2,a_i)}{2} + \cdots + w_{N_s} \frac{Q'_s(1,a_i) + Q'_s(2,a_i) + \cdots + Q'_s(N_s,a_i)}{N_s}$$

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

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, 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 high value. And the actions that are selected near the start state are evaluated as low value 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 should be positive, so the conversion of 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 and the negative value converted to the value from 0 to p. We can tune 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 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 coefficients $(w_1, \cdots, w_{N_s})$, we can set the ratio of the special actions to general actions. And in the proposed method, it is possible to set that the special actions and the general actions are evaluated

as a high credit simultaneously, so they can coexist. And our proposed method does not have "don't care" symbol that is used in classifier system (Holland, 1986), so the problems caused by using "don't care" symbol do not exist.

3.2.9 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

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

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

3.2.10 Fitness

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

$$fit(a_i) = fit_Q(a_i) + k_f \cdot fit_u(a_i) \tag{6}$$

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

3.2.11 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 method is used.

4 Application to Redundant Robot

In this section we apply QDSEGA to a hyper-redundant robot. We impose the same locomotion task to two different forms: multi-legged robot and snake-like robot, and we consider the ability that adapts control methods to differences of the body automatically.

4.1 Multi-legged robot

4.1.1 Task

The task is how to get closer to the light source. Fig. 9 shows the outline of the task. The light source is far enough from the start position of the robot and the reward is calculated using the distance between the current position of the robot and the light source.

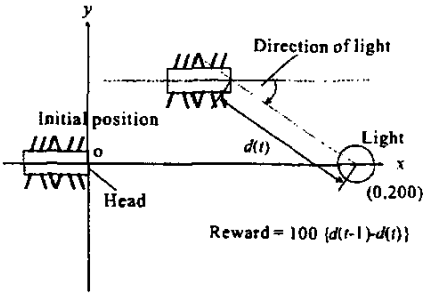

Figure 9: Task

4.1.2 Simulation Model

We consider 12-legged robot. We employ Minimal Simulation Model (MSM) that was proposed by M. Svinin et al., [8]. This model is very simple and it can be calculated very low computational cost.

Fig. 10 shows a multi-legged robot. Each leg has two joint and has four motoin and four states (1: Move forward and lift down, 2: Move back and lift down, 3: Move forward and lift up, 4: Move back and lift up). The position of the robot can be calculated by MSM as follows:

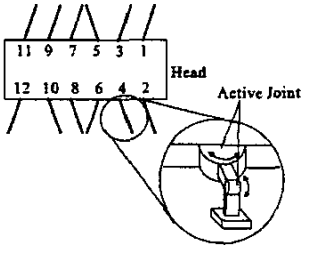

Figure 10: Multi-legged robot

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

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

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

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

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

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

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

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

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

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

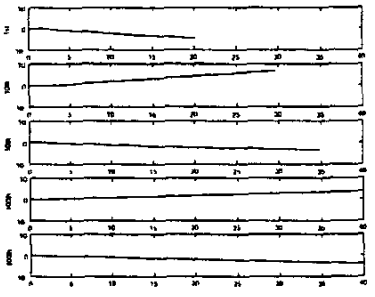

Figure 11: Trajectry of acquired locomotion of the multi-legged robot

where n_{ij}^r is the number of legs on the right side changing their configuration from the state i at the time t to the state j at the time $t+1$. Similarly define n_{ij}^l for the left side. F is total propulsion force and M is moment. And x,y,θ mean the position and orientation of the robot. Details are written in [8].

4.1.3 Simulation

<Formation of GA>

The dynamics of Genetic Algorithm of the proposed algorithm is composed as follows.

At first we describe the encoding. We define the action as the desired condition of legs. And the actions are encoded as the gene. One action expresses the all conditions of legs of one step. 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. And 200 times reproduction is carried out.

<Fomation of Q-learning>

The action space consists of the phenotypes of the generated genes. 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 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 by each step.

$$Reward = 100\{d(t-1) - d(t)\} \tag{17}$$

The roulette selection using the Boltzmann distribution is employed. The temperature of the Boltzmann distribution is given as follows.

$$T = 150.0 \times e^{(-0.002 \times trial)+1} \tag{18}$$

The learning rate is 0.5 and discounting rate is 0.9. The number of trials of each learning dynamics is taken as 10000.

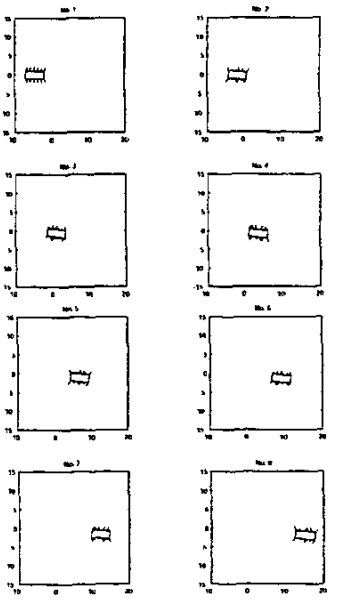

Figure 12: Acquired locomotion of the multi-legged robot

4.1.4 Simulation Results

Fig. 11 shows the trajectory of the robot and Fig. 12 shows the acquired locomotion at 200th generation. The number of the left side of Fig. 11 indicates the generations, the horizontal axis means the x-axis and the vertical axis means y-axis of Fig.9. From the Fig. 11 we find that as the generation increases, a movement distance is on the increase. In the simulation, the optimal movable distance is 40 so we can conclude that the acquired behavior is almost optimal.

4.2 Snake-like robot

4.2.1 Task

We consider a task which is the same as the subsection 4.1.1, using snake-like robot.

4.2.2 Simulation model of the snake-like robot

In this simulation we employ the dynamic model of the snake like robot with considering friction between robot body and environment proposed by Iwasaki et al [9]. All links touch the ground and the friction of the vertical direction with respect to the robot body is larger than that of the tangential direction. Owing to this difference of friction the snake-like robot can move. The number of links is 5. We regard that all actuator is stepping motor and the angle and angular velocity can be controlled. Lower agents move the corresponding joint to the desired angle with constant speed.

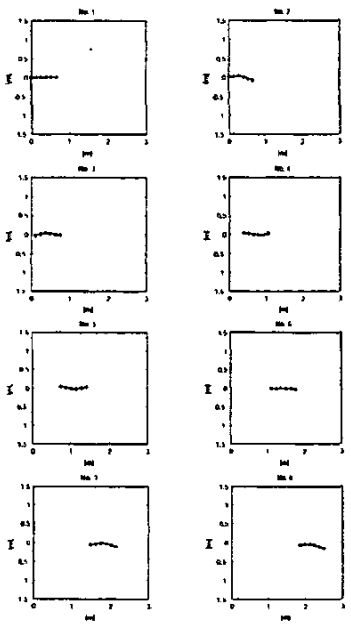

Figure 13: Acquired behavior (Acquisition of locomotion pattern by snake-like robot)

4.2.3 Simulation

<Formation of genetic algorithm>

Outline of the formation of genetic algorithm is same as the subsection 4.1.3. One gene has 9 characters that express the angles from -20[deg] to 20[deg] every 5 degrees. The number of individuals is 30. And roulette selection is employed. The probability of the crossover is 0.5 and one-point crossover is employed. The probability of mutation is 0.02. And 30 times reproduction is carried out.

<Formation of Q-learning>

Outline of formation of Q-learning is same as the subsection 4.1.3. The exterior state consists of the angle to the goal. The angle is divided four states (form -45 [deg] to 45 [deg] each 30 [deg]). 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. Reward is calculated as follows and it is given by each step.

$$Reward = 100\{d(t-1) - d(t)\} \tag{19}$$

4.2.4 Simulation Result

Fig. 13 shows the acquired behavior. We can find that the winding motion is acquired and the task is accomplished. It means that proposed algorithm is effective for not only the task in the static world but also the task in the dynamic world.

5 Conclusion

We have considered new approach that realizes flexible autonomous system. We regard a hyper-redundant robot as a flexibility of hardware and QDSEGA as flexibility of software. By combining these flexible mechanisms, we have realized flexible autonomous system. To demonstrate the effectiveness of the proposed approach, the locomotion task has been imposed to the multi-legged robot and snake-like robot. As a result effective locomotion patterns for each different bodies have been obtained automatically.

We can conclude that our proposed approach is effective to realize flexible autonomous systems.

Acknowledgements

This research is supported by The Grant-in Aid for COE Research Project of Super Mechano-Systems by The Ministry of Education, Science, Sport and Culture of Japan.

References

[1] T. Minato and M. Asada. Image feature generation by visio-motor map learning towards selective attention. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1422–1427, 2001.

[2] K. Yamada, K. Ohkura, M. Svinin, and K. Ueda. Adaptive segmentation of the state space based on bayesian discrimination in reinforcement learning. In *Proc. of the 6th Int. Symp. on Artificial life and Robotics*, pages 168–171, 2001.

[3] M. Yim, D. G. Duff, and K. Roufas. Modular reconfigurable robots, an approach to urban search and rescue. In *Proc. of 1st Int. Workshop on Human-friendly Welfare Robotics Systems*, pages 69–76, 2000.

[4] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, and S. Kokaji. Hardware design of modular robotic system. In *Proc. of 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages F-AIII-3-5 (CD-ROM), 2000.

[5] I. Erkmen, A.M. Erkmen, F. Matsuno, R. Chatterjee, and T. Kamegawa. Serpentine search robots in rescue operations. *IEEE Robotics and Automation Magazine*, 9(2), 2002. (to appear).

[6] 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.

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

[8] M. Svinin, S. Ushio, K. Yamada, and K. Ueda. Emergent systems of motion patterns for locomotion robots. In *Proc. of Int. Workshop on Emergent Synthesis*, pages 119–126, 1999.

[9] M. Saito, M. Fukaya, and T. Iwasaki. Serpentine locomotion with robotic snakes. *IEEE Control Syatems Magazine*, 22(1):64–81, 2002.