Engineering

Mechanical Engineering fields

Okayama University *Year* 2003

Extended QDSEGA for Controlling Real Robot : Acquisition of Locomotion Patterns for Snake : like Robot

Kazuyuki Ito
Okayama University

Tetsushi Kamegawa
Tokyo Institute of Technology

Fumitoshi Matsuno
Tokyo Institute of Technology

Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Extended QDSEGA for Controlling Real Robots
-Acquisition of Locomotion Patterns for Snake-like Robot-

Kazuyuki Ito[1], Tetsushi Kamegawa [2]and Fumitoshi Matsuno[2]

1) Department of Systems Engineering Okayama University
3-1-1, Tushima-naka, Okayama, 700-8530, Japan
e-mail kazuyuki@sys.okayama-u.ac.jp
2) Department of Computational Intelligence and Systems Science
Tokyo Institute of Technology
4259 Nagatsuta, Midori, Yokohama, 226-8502, Japan
e-mail {kamegawa, matsuno}@cs.dis.titech.ac.jp

Abstract

Reinforcement learning is very effective for robot learning. Because it does not need prior knowledge and has higher capability of reactive and adaptive behaviors. In our previous works, we proposed new reinforce learning algorithm: "Q-learning with Dynamic Structuring of Exploration Space Based on Genetic Algorithm (QDSEGA)". It is designed for complicated systems with large action-state space like a robot with many redundant degrees of freedom. However the application of QDSEGA is restricted to static systems.

A snake-like robot has many redundant degrees of freedom and the dynamics of the system are very important to complete the locomotion task. So application of usual reinforcement learning is very difficult.

In this paper, we extend layered structure of QDSEGA so that it becomes possible to apply it to real robots that have complexities and dynamics. We apply it to acquisition of locomotion pattern of the snake-like robot and demonstrate the effectiveness and the validity of QDSEGA with the extended layered structure by simulation and experiment.

1 Introduction

Reinforcement learning [1] is very effective for robot learning. It does not need prior knowledge, and has higher capability of reactive and adaptive behaviors. By applying reinforcement learning to the robot with many redundant degrees of freedom, adaptive autonomous system can be realized. So applying reinforcement learning to the robot with many redundant degrees of freedom is very attractive. However, there are some significant problems in applying it to them. Some of them are deep cost of learning and large size of action-state space. In the Q-learning [2], witch is one of the most famous and the most effective reinforcement learning algorithm, the size of Q-table increase exponentially with increase of degrees of freedom. So by using the conventional Q-learning, only a few degrees of freedom robots can be controlled.

In our previous works, we took it into consideration, and proposed new reinforcement learning algorithm: "Q-learning with Dynamic Structuring of Exploration Space Based on Genetic Algorithm (QDSEGA)[3, 4]". It is designed for complicated systems with large action-state space like a robot with many redundant degrees of freedom. In QDSEGA, Q-learning is applied to a small subset of exploration space to acquire some knowledge of a task, and then the subset of exploration space is restructured utilizing the acquired knowledge, and by repeating this cycle, effective subset and effective policy in the subset is acquired. So without prior knowledge, efficient search, compare to trial and error, is possible. By applying QDSEGA to the robot with many redundant degrees of freedom, an effective movement for each task is selected automatically from the various movements that can be realized by the redundancy of the robot. Effectiveness of QDSEGA and the adaptive autonomous systems were demonstrated using simulations of a 12-legged robot[4], and a 50-link manipulator[3]. However the applications of QDSEGA were restricted to static systems, and have never been applied to the real robots yet.

On the other hand, a snake-like robot is one of the most difficult examples of control problem. Because it has many redundant degrees of freedom and the dynamics of the system are very important to complete locomotion. So in the previous works of reinforcement learning, control of real snake-like robot have not been realized.

In this paper, we extend the layered structure of

0-7803-7736-2/03/$17.00 ©2003 IEEE

the QDSESA so that it can be applicable to the real robots with many redundant degrees of freedom. To demonstrate the effectiveness and the validity of the extended QDSEGA, we apply it to acquisition of locomotion pattern for the snake-like robot in simulations and experiments.

2 Proposed Algorithm

2.1 Outline

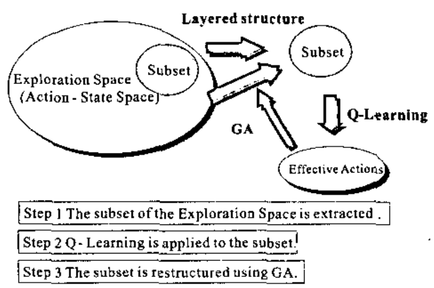
Layered structure
Subset
Exploration Space
(Action - State Space)
Subset
Q-Learning
GA
Effective Actions
Step 1 The subset of the Exploration Space is extracted .
Step 2 Q - Learning is applied to the subset.
Step 3 The subset is restructured using GA.

Figure 1: Outline of QDSEGA

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 utilizing the acquired knowledge. The reinforcement learning is applied to the new subset, and 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 utilizing the 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.

In our previous works[3, 4], we had assumed that the lower agents of the layered structure have enough ability to control each joint, and considered ideal static systems in the simulated world. In this paper, we improve QDSEGA to withdraw the assumption, and we consider the real robots that have dynamics, complexity and limited ability. To realize the improvements we extend the layered structure. Details are written as subsection 2.3.

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 state except for the interior state.

2.3 Extended Layered Structure

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

In our previous works, we had assumed that the lower agent has enough ability and each interior state converges to desired state within each time step. In this paper we withdraw the assumption and extend the layered structure.

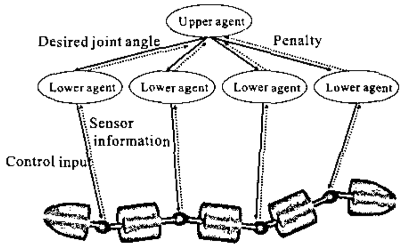

Figure 2: Hierarchal Structure

At first, we extend the conventional one-way communications between the upper agent and the lower agents to two-way communication.

Next we extend the lower agents to return some information to the upper agent. 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.

Next, we extend the upper agent to catch the penalties form the lower agents. 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 the 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 above improvements, extended QDSEGA 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, and our improvements make it possible to apply QDSEGA to real lower agents that have limited ability.

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, 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 any interior state that can be transited by any action in the subset is 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, 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.

We can structure the subset of exploration space dynamically by structuring the action space dynamically. In QDSEGA, the actions are structured using genetic algorithm in the learning process of the upper agent.

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

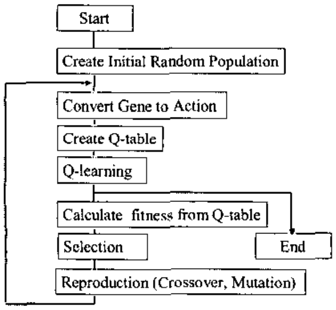
Start
Create Initial Random Population
Convert Gene to Action
Create Q-table
Q-learning
Calculate fitness from Q-table
Selection
End
Reproduction (Crossover, Mutation)

Figure 3: Learning process of the upper agent

2.6 Encoding

In this algorithm, each individual expresses the selectable action on the learning dynamics. 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 restructured using Genetic Algorithm. The number of individuals means the size of the subset.

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. 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[2] 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.

2.9 Fitness

2.9.1 Fitness of Q-table

The fitness of genes 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.

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

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 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 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 which decides the ratio of special actions to general actions.

2.9.2 Fitness of frequency

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 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.

2.9.3 Fitness

Combining discussion in the subsections 2.9.1, 2.9.2 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.

2.10 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 the conventional method is used.

3 Acquisition of Locomotion Patterns

In this section, we apply the proposed method to acquisition of locomotion pattern for snake-like robot. At first the learning process is carried out in the simulated world, and then acquired locomotion is applied to the real robot to demonstrate the validity of the acquired behavior in applying to the real robot which has dynamics and complexity.

3.1 Snake-like robot

We employ real snake-like robot [5] that consist of 5 links that can move in the same horizontal plane.

Fig. 4 shows the snake-like robot. Each joint has stepping motor to drive the link. A passive wheel is attached to the bottom of each link. The wheel can rotate to the direction that is parallel to the link. So the friction to the parallel direction is smaller than that of rectangular direction. By winding the body suitably, the snake-like robot can move using the differences of friction. The size of each link is 80[mm] × 80[mm] × 130[mm], and weight is 600[g].

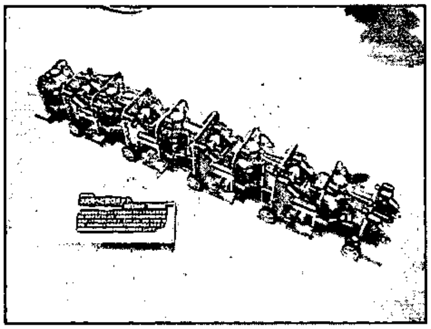

Figure 4: Snake-like Robot

3.2 Task

The task is how to get closer to the goal. Fig. 5 shows the outline of the task. The goal is far enough from the start position and the reward is calculated using the distance from the goal.

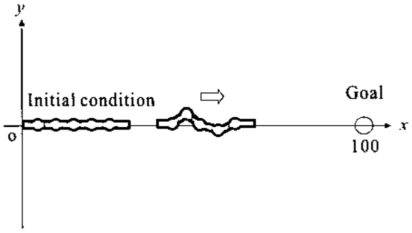

Figure 5: Task

3.3 Simulation model

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 [6].

The model of stepping motors and lower agents are realized as follows. At first the constant angler velocity $\dot{\theta}(t_{i+1})$ of each joint that realizes the desired state within each time step is calculated from the differences of the current state $\theta(t_i)$ with the desired state $\theta_d(t_{i+1})$.

$$\dot{\theta}(t_{i+1}) = \frac{\theta(t_i) - \theta_d(t_{i+1})}{t_{i+1} - t_i}$$

Next we take the ability of the stepping motor into consideration, we consider two cases. One is the case when the calculated angular velocity is enough small compare to the ability of the stepping motor. In that case, we regard that each joint move to the desired state by the calculated angular velocity. In the other case, this means that the calculated angular velocity is too large to move the robot by the stepping motor, we regard that the stepping motor can not move and a penalty is returned to the upper agent.

3.4 Simulation

<Formation of genetic algorithm>

The dynamics of GA of the proposed method is composed as follows. At first we describe the encoding. We define the action as the desired angles of the joints. Fig. 6 shows the encode method. One action expresses all joint angles of one sharp of the snake-like robot. One action is encoded one chromosome and the chromosome has a same number of genes as the number of joints. One gene expresses the angle of one joint. 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 uniform crossover is employed. The probability of mutation is 0.02. And 30 times reproduction is carried out.

<Formation of Q-learning>

The action space consists of the phenotypes of the generated genes. The state space consists of the initial state and the states that can be transited by generated actions. 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. Where $d(t)$ is a distance between the robot and the goal in step t.

$$Reward = 100\{d(t-1) - d(t)\} \qquad (7)$$

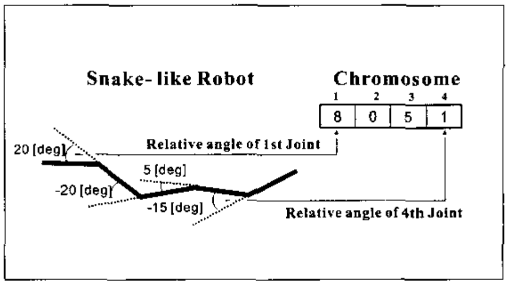

Figure 6: Encode method

3.5 Simulation Result

Fig. 7 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.

The locomotion task is a same as that of our previous works of the multi-legged robot [4]. We can also find that the different suitable behaviors for each different body are emerged by the same algorithm. It means that the QDSEGA has autonomy, flexibility and adaptability.

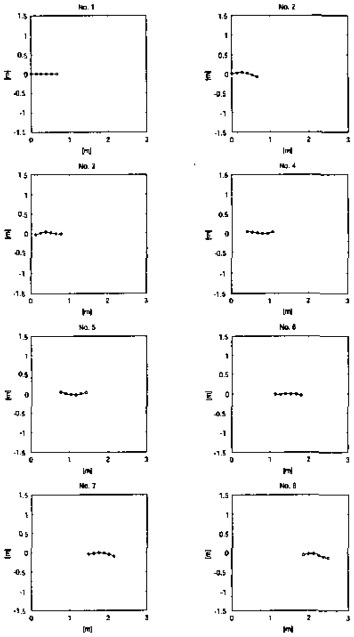

Figure 7: Acquired behavior

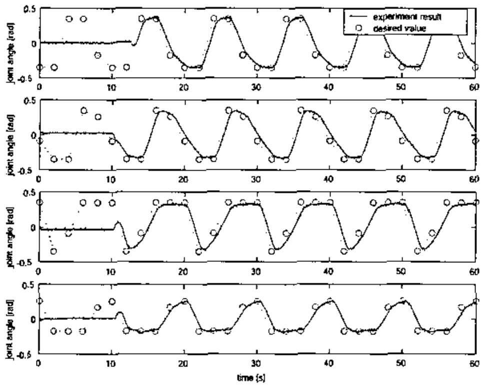

Figure 8: Transient Responses

4 Experiment

Experiments have been carried out to demonstrate the effectiveness and the validity of the acquired behaviors. The acquired locomotion patterns in the simulation are implemented to the snake-like robot.

Five different patterns are implemented and all patterns could be realized by the real robot and the locomotion is completed. Fig. 7 shows the transient responses of each joint. The circle in the Fig. 7 means the desired state that is acquired by the learning process of the upper agent, and the dotted line means the desired joint angle that is realized by the lower agent, and the line means transient responses of real robot.

We can find that the joint angles of the real robot converge to the desired values that are acquired by the proposed learning architecture. It means that the acquired behavior consist of only possible actions and we can conclude that the two-way communication of the layered structure is valid and the proposed algorithm is applicable for real robot that have actuators with limited ability.

Fig. 9 shows the realized locomotion by the real robot. We can find that the winding motion is realized and the task is accomplished. It means that the proposed algorithm is effective for not only idealized simple systems in the simulated world but also complicated system in the real world.

5 Conclusion

In this paper we proposed the new reinforcement learning algorithm for the real robots with many degrees of freedom by extending QDSEGA. To demonstrate the effectiveness of the proposed algorithm, simulations of acquisition of locomotion patterns for a snake-like robot have been carried out, and the acquired locomotion have been applied to the real snake-like robot. As a result, winding motions have been acquired automatically and the locomotion has been realized by the real robot. We can conclude that the proposed algorithm is effective for not only idealized simple systems in the simulated world but also complicated system in the real world.

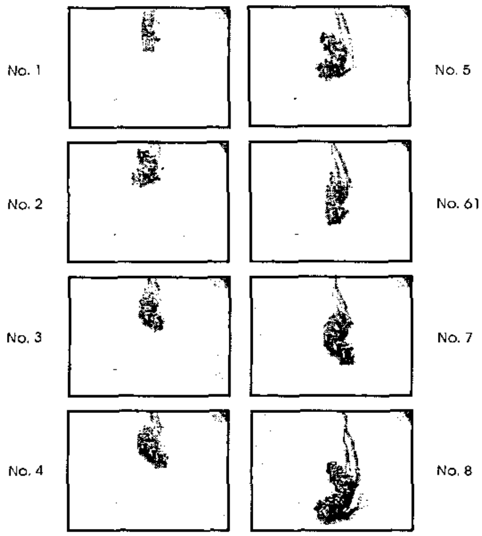

Figure 9: Realized behavior

References

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

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

[3] 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.

[4] 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.

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