The Fourteenth International Symposium on Artificial Life and Robotics 2009 (AROB 14th '09),
B-Con Plaza, Beppu, Oita, Japan, February 5 - 7, 2009

Autonomous Reconfiguration of Robot Shape by Using Q-learning

Satoshi Shiba[*1] Masafumi Uchida[*1] Akio Nozawa[*2] Hirotoshi Asano[*3] Hitoshi Onogaki[*4]
Tota Mizuno[*5] Hideto Ide[*3] and Syuichi Yokoyama[*4]
*1 The University of Electro-Communications, 1–5–1 Chofu-ga-oka, Chofu, Tokyo 182–8585
*2 Meisei University, 2–1–1, Hodokubo, Hino, Tokyo 191–8506
*3 Aoyama Gakuin University, 5–10–1, Fuchinobe, Sagamihara, Kanagawa 229–8558
*4 Kogakuin University, 1–24–2, Nishi-shinjuku, Shinjuku-ku, Tokyo 163–8677
*5 National Institute of Advanced Industrial Science and Technology (AIST),
Central6, 1–1–1 Higashi, Tsukuba, Ibaraki 305–8566

Abstract

A modular robot can be built with a shape and function that match the working environment. We developed a four-arm modular robot system which can be configured in a planar structure. A learning mechanism is incorporated in each module constituting the robot. We aim to control the overall shape of the robot by accumulation of the autonomous actions resulting from the individual learning functions. Considering that the overall shape of the modular robot depends on the learning condition in each module, this control method can be treated as a dispersion control learning method. The learning object is the cooperative motion between adjacent modules. The learning process proceeds based on Q-learning by trial and error. We confirmed the effectiveness of proposed technique by computer simulation.

1 Introduction

Most of traditional robots have been built to perform particular tasks in place of humans. In future, however, robots are expected to be able to perform a wide range of tasks autonomously. Conventional robots are limited to performing just one or two tasks assumed by the designer. It is almost impossible for a single robot to adapt to various kinds of tasks and environments. To overcome this limitation, a modular robot was proposed as a system that can be adapted to various given tasks and unknown environments [1, 2, 5, 6]. A modular robot can be defined as a robotic system constructed from a set of standardized components, so-called "modules". With this approach, the robot body is reconfigured for various complex tasks, instead of designing a new, different robot for each task. Modules, by themselves, cannot perform tasks, but when many of them are connected together, a new system can be obtained to do complicated tasks.

A modular robot changes its shape by changing the connections between modules in order to meet the demands of different tasks or different working environments. Over the last ten years, research in this field has focused on versatility and adaptability aspects, but less effort has been made in the field of self-reconfigurable modular robots that can autonomously change their configuration [10, 11, 12].

In this study, we developed a multi-arm modular robot, and we propose a system that incorporates a learning mechanism of each module constituting the modular robot. We aim to control the overall shape of the modular robot by the accumulation of autonomous actions arising from the learning mechanism of each module. Since the overall shape of the modular robot depends on the learning result of each module, this method can be treated as a dispersion control learning method [7, 8, 9]. The cooperation between a module and an adjacent module is the object of learning. The learning process involves trial and error based on Q-learning [13, 14]. We confirmed the effectiveness of the proposed technique by computer simulation.

2 Function of a module and Q-learning

Figure 1 shows a model of the four-arm module and the assumed modular robot constructed from such modules [3, **?**]. Each arm has the ability to be connected to and disconnected from other arms of the module. The shape of the robot can be reconfigured by changing connections with adjacent modules according to preset rules, as shown in Figure 1. Here,

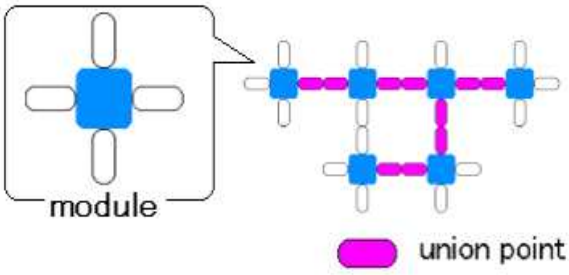

Figure 1. Model of four-arm module and modular robot.

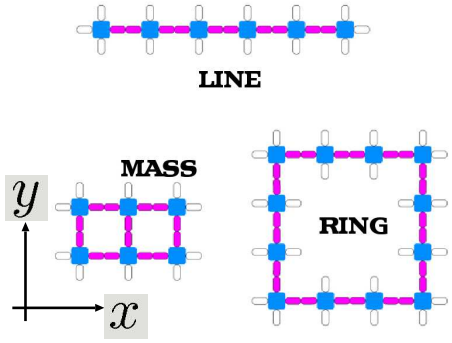

Figure 2. Three robot shapes considered (**LINE**, **MASS** and **RING**).

the connection state of an arm is expressed by two values (binary) as follows: connected ($s = 1$), not connected ($s = 0$). Since there are four arms per module, the connection states of the arms in one module can be expressed by four bits. The following four basic operations are defined for each arm: (i) arm is turned clockwise [**MOVE CW**], (ii) arm is turned counter-clockwise [**MOVE CCW**], (iii) arm is connected [**CONNECT**], and (iv) arm is disconnected [**DISCONNECT**]. These four basic actions cannot be performed by a single module; they are performed by the cooperative behavior of two or more modules. An additional standby action [**STAY**] is also added. As a result, there are 17 kinds of action **a** in total: 4 arms × 4 basic actions + 1 standby action. A module acts in accordance with the rule of state **s** versus action **a**. The appearance probability of an action **a** in the state **s** is optimized by Q-learning.

Figure 2 shows three robot shapes that we consider. They are called the **LINE** shape, **MASS** shape, and **RING** shape. Since the action of each module is decided by Q-learning, the overall shape of the robot is controlled by Q-learning. The reconfiguration process of the robot shape was examined by computer simulation.

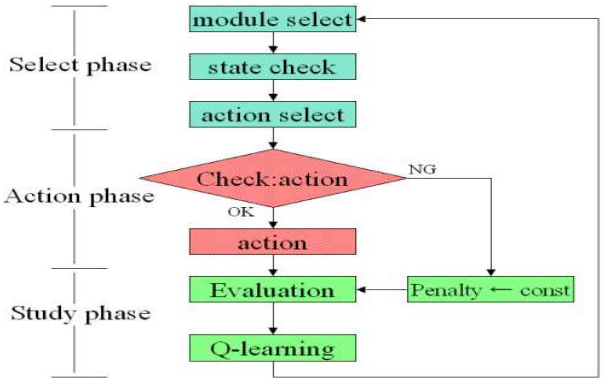

Figure 3. Flow of learning algorithm.

To develop a system in which the shape of the robot is automatically reconfigured, each module proceeds with Q-learning while repeating trial and error actions in the early stage. This Q-learning is not performed uniformly for the entire robot, but is performed independently in each module.

Figure 3 shows the flow of the learning algorithm. First, one module is chosen at random from the group of modules constituting the robot. The chosen module decides an action in accordance with its own Q value. After checking whether the action is valid, the module carries out the action. Next, the action is evaluated, and the Q value is updated in accordance with the evaluation result. This series of processes is defined as one step". The number of modules that act in one step is assumed to be only one. According to an observed state $\mathbf{s}_t$ at time t, an action $\mathbf{a}_k$ for the module is selected with probability $P(s_t, a_k)$ presented in Eq. (1) below through a Boltzmann selection method. If it is judged to be an invalid action, it is not carried out. In that case, this action is dealt with by imposing a penalty value on the Q value. In Eq. (1), $ANUM$ is the total number of actions, and $Q(s_t, a_k)$ is an expectation value of action $\mathbf{a}_k$ $(k = 1, 2, \cdots, ANUM)$ in a state $\mathbf{s}_t$. T is the Boltzmann temperature presented in Eq. (2). In Eq. (2), τ is a time constant.

$$P(\mathbf{s}_t, \mathbf{a}_k) = \frac{e^{\frac{Q(\mathbf{s}_t, \mathbf{a}_k)}{T}}}{\sum_{k=1}^{ANUM} e^{\frac{Q(\mathbf{s}_t, \mathbf{a}_k)}{T}}} \tag{1}$$

$$T = 2 \cdot e^{-\frac{t}{\tau}} \tag{2}$$

Then, $Q(s_t, a_k)$ is updated as shown in Eq. (3) according to an obtained reward *Reward* by carrying out an action based on Eq. (1).

$$Q(\mathbf{s}_t, \mathbf{a}_k) \leftarrow (1-\alpha) \cdot Q(\mathbf{s}_t, \mathbf{a}_k) + \alpha \left\{ Reward + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}_{t+1}, \mathbf{a}') \right\} \quad (3)$$

$$Reward = \frac{Eval_{t-1} - Eval_t}{Eval_t + 1} - Penalty \quad (4)$$

Here, α is a learning rate, γ is a discount rate, $\max_{a'} Q(s_{t+1}, a')$ is a maximum expectation value in state $\mathbf{s}_{t+1}$ observed at the next time step, and $\mathbf{a}'$ is an action indicated by this maximum expectation value. In Eq. (4), *Penalty* is a penalty value, and $Eval_t$ is an evaluation value of the robot shape at time t.

3 Computer Simulation

We simulated the reconfiguration of the modular robot shape from an initial shape to (i) the **LINE** shape and (ii) the **MASS** shape. The initial shape of the robot was set so that modules formed a chain. The evaluation value is $Eval_t$ when the shape of the robot is evaluated from the viewpoint of an outside observer at time t. First, the shape of the robot was estimated from the positions of all modules in the simulation. Next, $Eval_t$ was calculated from the difference between the estimated shape and the requested shape (**LINE** and **MASS**). In the case of the **LINE** shape, the deviation on either side of the x axis or the y axis is the evaluation value $Eval_t$ concerning the position of all modules. On the other hand, in the case of the **MASS** shape, the deviation of the position of all modules is the evaluation value $Eval_t$.

Minimizing $Eval_t$ is equivalent to achieving the requested shape in both cases. The learning process proceeds by comparing the evaluation value after an action with the evaluation value before the action.

The convergence time in learning and the required memory storage capacity depend on the size of the learning space, calculated by the product of the total number of states **s** and the total number of actions **a** in the Q-learning. We considered two ways of reducing the learning space to improve the learning efficiency. The first way was to exclude the choice of the arm from the learning object, reducing the number of possible actions from 17 to 5. The learning space was reduced by 70 % as a result. We called this the "action only" method. The second way is to pair the [**CONNECT**] action in one module and the [**DISCONNECT**] action in an adjacent module and to unify them by introducing a new [**REVERSE**] action. As a result, the total number of actions decreased from 17 to 7, and the learning space was reduced by 59 %. We called this the "new model" method.

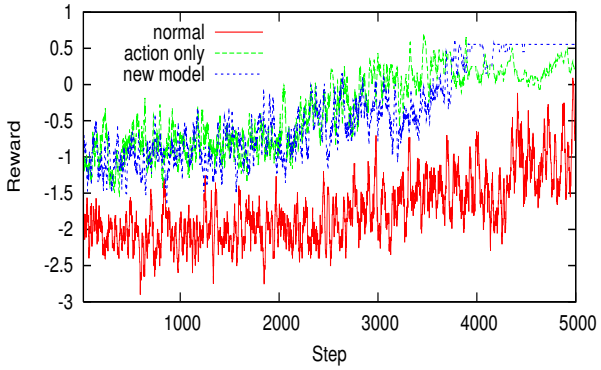

Figure 4. Comparison of 5,000 learning steps (**LINE** shape).

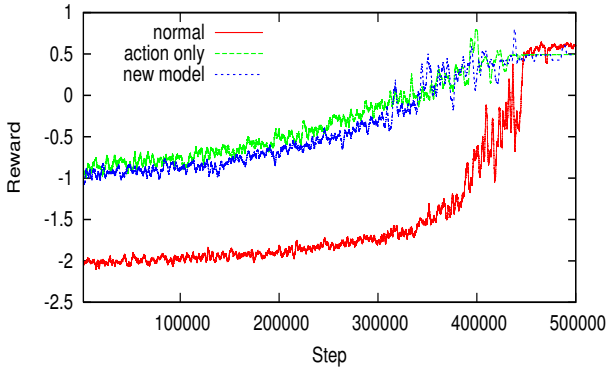

Figure 5. Comparison of 500,000 learning steps (**LINE** shape).

Figures 4 and 5 show the simulation results of reconfiguration based on the evaluation value of the **LINE** shape in a robot body composed of 10 modules. In these two figures, the solid line indicates the results obtained without using either of the methods described above ("normal"). The dotted line indicates the "action only" type, and the broken line indicates the "new model" type. The horizontal axis is the number of learning steps, and the vertical axis is the mean value of *Reward* for each module. In Figures 4 and 5, the upper limit was set to 5,000 steps and 500,000 steps, respectively. The "action only" type converges earlier to a certain value at the end of the learning step.

As shown in Figure 4, the amount of change in *Reward* with the "normal" type is large in the ini-

tial stage of the learning process, and the final convergence is insufficient. On the other hand, the "action only" type has a more expensive *Reward* at the early stage,and *Reward* becomes a constant, stable value at the last stage, showing successful convergence. The "new model" type shows the best result in terms of the rate of rise in *Reward* at the last stage and the stability after reaching convergence. For the "action only" type, the learning object was 5 actions. Therefore, it determines that learning is completed in fewer steps, but there is no optimization related to the choice of arm because arm choice was excluded from the learning object. Therefore, the final convergent value does not reach the optimum value. The learning space of the "new model" type is larger than that of the "action only" type. However, because the choice of arm is contained in the learning object of the "new model" type, the final convergent value reaches the optimum value. As shown in Figure 5, if a sufficient number of learning steps for all three types is ensured, it is possible to achieve a final convergent result that is equivalent to that of the "new model" type. Figure 6 shows one example of the **LINE** shape and the **MASS** shape after learning with the "new model" type. In this figure, the blocks with numbers represent modules (numbers $0, 1, 2, \cdots$ are modules' IDs), and the "**" mark indicates the connections between modules. The resulting shape is not necessarily always optimized. However, we confirmed that the shape of the robot could be controlled to a certain extent without using communication between modules.

Figure 6. **LINE** shape (left) and **MASS** shape (right) obtained by using the "new model" type.

The evaluation value to form the **RING** shape was defined as follows. The requested radius of the **RING** shape is D_0. An arbitrary module is M. A module adjoining M is M_1. The furthest module from M is M_2. An evaluation value $Eval_t$ is defined by Eq. (5) based on a distance D between the position at distance

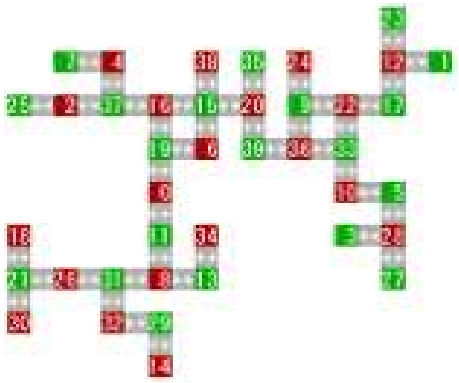

Figure 7. Reconfiguration result for **RING** shape ($D_0 = 9.0$).

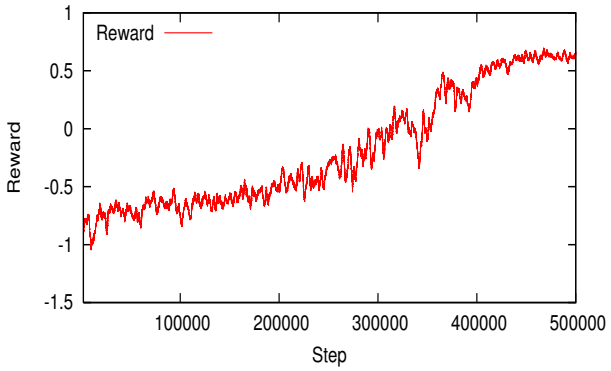

Figure 8. Evolution of *Reward* value in reconfiguration of the **RING** shape ($D_0 = 9.0$).

D_0 and the position of M from the midpoint of M_1 and M_2.

$$Eval_t = e^{(D_0 - D)^2} \tag{5}$$

This rule was established based on the circular alignment algorithm of multi-robot systems[15].

Figures 7 and 8 show the simulation results of reconfiguration based on the evaluation value of the **RING** shape. In this simulation, we specified $D_0 = 9.0$ with 40 modules used, and the number of learning steps was set to 500,000. In Figure 8, the horizontal axis is the learning step number, and the vertical axis is the *Reward* value. As shown in this figure, the *Reward* value increased as the step number increased, showing that the learning function proceeded. However, the modules did not fully converge to the *RING* shape. Figures 9 and 10 show the evolution of the obtained *Reward* value of each module constituting the robot. Figure 9 shows the case of the **LINE** shape,

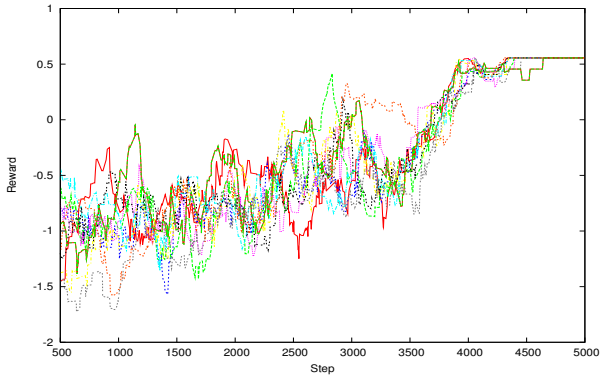

Figure 9. Comparison of evolution of *Reward* value for each module in the case of the **LINE** shape.

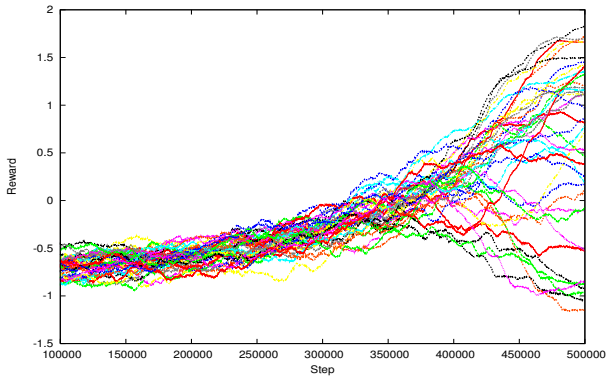

Figure 10. Comparison of evolution of *Reward* value for each module in the case of the **RING** shape ($D_0 = 9.0$).

and Figure 10 shows the case of the **RING** shape with $D_0 = 9.0$. The obtained *Reward* value of all modules tends to increase in the case of the **LINE** shape. The final convergent value for the obtained *Reward* is the same for all modules. Although there was a difference in the obtained *Reward* between the modules in the first half of the learning process, this difference gradually reduced in the latter half. On the other hand, in the case of the **RING** shape, convergence could not be confirmed from the obtained *Reward* between each module in the learning process. The obtained *Reward* values for each module diverged in the latter half of the learning process. Comparing Figure 9 with Figure 10, we found that, in order to obtain a good result, it is necessary to reduce the differences in the obtained reward among the modules. In the learning result shown in Figure 10, the module which converged to the highest obtained *Reward* was ID:7, and the module which converged to the lowest obtained *Reward* was ID:16. Figures 11 and 12 show the number of invalid actions, determined based on the Q value, in module ID:7 and module ID:16, showing the states after 100,000 steps and after 500,000 steps, respectively. In each figure, the left part shows module ID:7, and the right part shows module ID:16. The horizontal axes of the figures show the state of the module, and the vertical axes show action 0:[**STAY**], 1:[**MOVE CW**] for arm No. 1, 2:[**MOVE CCW**] for arm No. 1, 3:[**MOVE CW**] for arm No. 2, 4:[**MOVE CCW**] for arm No. 2, 5:[**REVERSE**] for arm No. 3, and 6:[**REVERSE**] for arm No. 4. Here, arm Nos. 1–4 correspond to the four arms of a module. This figure shows the number of invalid actions that occurred in the first 5,000 steps. The actions in the state indicated by the thick red part shows actions that were often judged invalid.

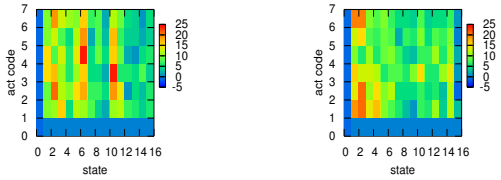

Figure 11. The total number of invalid actions in 100,000 steps for ID:7 (left) and ID:16 (right).

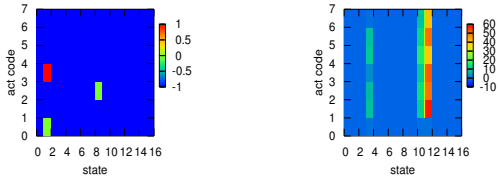

Figure 12. The total number of invalid actions in 500,000 steps for ID:7 (left) and ID:16 (right).

It was confirmed that the number of actions judged invalid gradually reduced as the learning process proceeded. There were more actions judged invalid in the module with ID:16 than in the module with ID:7 in the latter half of the learning process. In other words, the learning process for the module with ID:16 was delayed. An action judged invalid is an action that cannot be carried out in the actual environment. Therefore, when an invalid action is output, the learning process cannot proceed. As a result, the learn-

ing process of the module that outputs more actions judged invalid is delayed. It is possible that this problem might be improved by making the learning process proceed equally.

4 Conclusion

In this research, we examined the reconfiguration control of the shape of a modular robot. A robot composed of four-arm modules was considered as a model. We proposed providing each module with a learning function based on Q-learning. We verified the validity of the proposed technique by computer simulation. Our results showed that, when learning of each module was integrated in the overall robot, the requested shape of the robot could be realized.

Problems that we currently face in advancing our work include: (1) increasing the number of modules; (2) expanding the approach to a 3-D environment; and (3) verifying the performance in real environments.

Acknowledgements

This work was supported in part by Grant-in-Aid of Hayao Nakayama Foundation for Science & Technology and Culture, Japan. This support is gratefully acknowledged.

References

[1] T. Fukuda and S. Nakagawa: "Dynamically Reconfigurable Robotics System," Proc. of IEEE Int. Conf. Robotics and Automation, pp.1581–1586(1988)

[2] G. S. Chirikjian: "Metamorphic Hyper-Redundant Manipulators," Proc. of JSME Int. Conf. on Advanced Mecatronics, pp.467–472(1993)

[3] K.Kotay and D. Rus: "Locomotion versatility through self-reconfiguration," Robotics and Autonomous Systems, vol.26, pp.217–232(1999)

[4] D. Rus and M. Vona: "Crystalline robots: Self-reconfiguration with unit-compressible modules," Autonomous Robots, vol.10, no.1, pp.107–124(2001)

[5] M. Yim: "New locomotion gaits," Proc. of IEEE ICRA, pp.2508–2514(1994)

[6] C. Paredis, H.B. Brown and P. Khosla: "A rapidly deployable manipulator system," Proc. of IEEE ICRA, pp.1434–1439(1996)

[7] W.-M.Shen, B.Salemi and P.Will: "Hormone-Inspired Adaptive Communication and Distributed Control for CONRO," IEEE Trans. on Robotics and Automation, Vol.18, No.5, pp.700–712 (2002)

[8] J. Walter, J. Welch and N. Amato: "Distributed Reconfiguration of Metamorphic Robot Chains," Proc. of the 19th Annual ACM SIGACTSIGOPS Symposium on Principles of Distributed Computing (PODC 2000), Portland, Oregon, pp.171–180(2000)

[9] Z. Bulter, K. Kotay, D. Rus and K. Tomita: "Cellular Automata for for Decentralized Control of Self-Reconfigurable Robots," Proc. of the 2001 IEEE Int. Conf. of Robotics and Automation workshop on Robotics Modular Robots (2001)

[10] A. Pamecha, I. Ebert-Uphoff and G. Chirikjian: "Useful Metric for Modular Robot Motion Planning," IEEE Trans. on Robotics and Automation, vol.13, no.4, pp.531–545 (1997)

[11] D. Rus and M. Vona: "Crystalline Robots: Self-Reconfiguration with Compressible Unit Modules," Autonomous Robots, vol.10, no.1, pp.107–124 (2001)

[12] C.-H. Chiang and G. Chirikjian: "Modular robot motion planning using similarity metrics," Autonomous Robots, vol.10, no.1, pp.91–106, (2001)

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

[14] R.S. Sutton and A.G.Barti: "Reinforcement Learning: An Introduction," MIT Press(1998)

[15] Koji Ito: "Intelligence and Emergence," NTT Publishing Co., Ltd.,pp.60-70 (2000)