Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Robotic action acquisition with cognitive biases in coarse-grained state space

Daisuke Uragami[a,*], Yu Kohno[b], Tatsuji Takahashi[c]

[a] College of Industrial Technology, Nihon University, 1-2-1, Izumi, Narashino, Chiba, 275-8575, Japan
[b] Graduate School of Advanced Science and Technology, Tokyo Denki University, Hatoyama, Hiki, Saitama, 350-0394, Japan
[c] School of Science and Technology, Tokyo Denki University, Hatoyama, Hiki, Saitama, 350-0394, Japan

ARTICLE INFO

Article history:
Received 29 March 2016
Received in revised form 14 May 2016
Accepted 15 May 2016
Available online 16 May 2016

Keywords:
Loosely symmetric model
Q-learning
Acrobot
Giant-swing robot
Partially observable markov decision process
Biologically inspired cognitive architecture

ABSTRACT

Some of the authors have previously proposed a cognitively inspired reinforcement learning architecture (LS-Q) that mimics cognitive biases in humans. LS-Q adaptively learns under uniform, coarse-grained state division and performs well without parameter tuning in a giant-swing robot task. However, these results were shown only in simulations. In this study, we test the validity of the LS-Q implemented in a robot in a real environment. In addition, we analyze the learning process to elucidate the mechanism by which the LS-Q adaptively learns under the partially observable environment. We argue that the LS-Q may be a versatile reinforcement learning architecture, which is, despite its simplicity, easily applicable and does not require well-prepared settings.

1. Introduction

In order to realize autonomous robots, in addition to the hardware made possible by the progress in control engineering, we need appropriate software that can learn through trial-and-error without preliminary programming. Machine learning techniques represented by representation learning (LeCun et al., 2015) enable embodied machines to become far more autonomous, supported by fast learning and high generalization from a limited number of examples., When there is uncertainty, or no prior knowledge, autonomous learning of robot body control is realized by reinforcement learning (RL). This type of learning involves an agent attempting to determine an action sequence that maximizes the total reward with the clue of rewards given by the interaction between the agent and the environment (Sutton and Barto, 1998).

In a classical RL algorithm, we define a state and/or action value function over discrete states and update the function based on finite trials. Contrarily, the state space of the real environment of the robotic agent is typically continuous. Hence, the granularity of the discretized state space or the function approximation methods is the crucial factor in learning. The deep Q-network (DQN) is a recent successful algorithm equipped with a function approximation module that is based on a fine-grained state representation (Mnih et al., 2015a).

In this study, we focus on robotic action acquisition over a coarse-grained state space, without function approximation. RL in coarse-grained state spaces is an important research topic in terms of simplification and increasing speed (Munos and Moore, 2002; Ormoneit and Sen, 2002). In addition, environment identification under coarse-grained state division is a major topic in cognitive robotics of the emergence of symbols (Barsalou, 1999; Tani, 1996; Taniguchi et al., 2016).

In terms of the learning agent, the coarse-grainedness of the state space is a source of uncertainty in the environment. One type of architecture that has been proven to be effective is bio-inspired (Genewein et al., 2015; Niizato and Gunji, 2011; Pfeifer et al., 2007) which mimics some adaptive aspects of living systems. In this study, we exploit a cognitive bias model in human beings called the loosely symmetric (LS) model (Kohno and Takahashi, 2015; Oyo et al., 2015; Shinohara et al., 2007; Takahashi et al., 2010). The LS model implements a symmetric valuation of relationships that is considered applicable to human beings (Sidman et al., 1982), a relative evaluation of actions derived from a mutual exclusivity (Markman and Wachtel, 1988), and *satisficing* behavior that is a representative strategy under bounded rationality (Simon, 1956). The LS model precisely describes the manner in which human intuition in a causal relationship works given the co-occurrence

* Corresponding author.
E-mail addresses: dduragami@gmail.com (D. Uragami), yu.kohno02@gmail.com (Y. Kohno), tatsuji.takahashi@gmail.com (T. Takahashi).

http://dx.doi.org/10.1016/j.biosystems.2016.05.007
0303-2647/© 2016 Elsevier Ireland Ltd. All rights reserved.

information between two events, a candidate cause and the effect in focus (Oyo et al., 2015). It is further shown that an agent equipped with a valuation of actions by the LS model (as a value function in reinforcement learning) performs better than the standard algorithms for a broad range of tasks in *K*-armed bandit problems, which form the most fundamental class in reinforcement learning (Takahashi et al., 2011). Oyo and others analyzed the LS in terms of the framework of empirical Bayes in terms of its cognitive properties (Oyo et al., 2015). However, the application range of the LS model was somewhat limited due to the fact that it allows only two actions and two levels of outcome for the actions. Kohno and Takahashi analyzed the LS model from a viewpoint of satisficing and generalized it to have an arbitrary number of actions, and arbitrary satisficing criterion (aspiring level). In addition, it was found to perform at a higher level than UCB1-tuned (Auer et al., 2002), a standard algorithm for bandit problems (Kohno and Takahashi, 2015).

For a class of adaptive tasks broader than bandit problems, Uragami and others proposed a method of applying the LS model to the general reinforcement learning algorithm Q-learning, LS-Q (Uragami et al., 2014). The LS-Q algorithm can be applied to general reinforcement learning tasks with an arbitrary number of states, whereas there is only one state in bandit problems. LS-Q has been applied to motion acquisition of a simulated robot. It was confirmed that (a) it effectively learns the action sequences under coarse-grained state divisions, and (b) this learning is effective for various parameter, including the learning rate and length of learning. In the case of (a), it is usually necessary for a reinforcement learning agent to reflect the dynamics of the environment effectively in the setting to work (Sutton, 1996). However, as it will be described, LS-Q does not presuppose any specific knowledge of the environment, defining the state space in the simplest way. As for (b), it is suggested that the LS-Q model is an architecture that can adapt to various environments without specific parameter tuning. However, these results were confirmed only in simulations and not in actual robots. In addition, although it was suggested that the LS-Q model can escape from stagnant loops in the motion acquisition, the mechanism that enables this has not yet been elaborated. Generally speaking, if the continuous state space is coarse-grained into a rough discrete state space, because of the incomplete state recognition, the environment becomes only partially observable and the assumptions required for classical reinforcement learning algorithms, such as Q-learning, to guarantee appropriate learning behavior are discarded (Kaelbling et al., 1998). If LS-Q can be applied to such an environment, its effectiveness will be increased. In this study, we build an actual robot for giant-swing applications and test if LS-Q is effective for real robots, with the aid of a simulator. We confirm that the environment is a partially observable Markov decision process, (POMDP) and examine its effect on the learning. We analyze the mechanism that enables LS-Q to adaptively learn in such an environment.

2. Methods

2.1. Acrobot (Giant swing robot)

A robot hung on a horizontal bar, giant swing robot, is also known as an acrobot. The action goal is to swing around the bar. It is often used as a test task for reinforcement learning methods (Hauser and Murray, 1990; Spong, 1995; Sutton and Barto, 1998). One of the characteristics of the acrobot is that only the lumbar joint is active (see Fig. 1). It is a nonholonomic system with less controllable dimensions than the total degrees of freedom. The issue for learning is to determine when and how to expand and contract the lumbar joint timeously. It is a simple yet nonlinear dynamic system, where the degree of uncertainty of the environment can be parameterized according to the resolution of the state division. In this sense, the acrobot is an adequate system that may bridge well-defined and well-behaved testbeds, and more realistic complicated subjects.

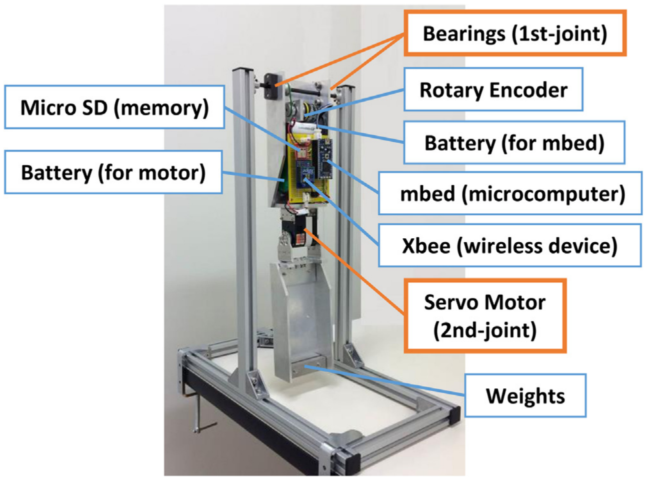


Fig. 1. Giant-swing robot. The first joint between the bar and the robot is free to move with bearing. The second joint between the first (upper body) and second links (lower body) actively operates with a servomotor. With the principal parts on the first link, the robot can rotate without tangling cables.

We built a real acrobot as shown in Fig. 1. The frames for the posts supporting the horizontal bar and the robot body are manufactured from an aluminum frame by MISUMI. The first joint, which is free to move, between the robot and the bar uses ball bearings to mitigate friction. The turning angle of this joint is measured by a rotary encoder. The second joint is a lumbar type that is controlled by a servo motor and connects the first link (upper body) and the second link (lower body). The motor is a KONDO serial servo motor (KRS6003HV). Through a serial connection, we control the rotation angle and velocity and measure the current rotation angle with the motor. We use a microcomputer board (*mbed*) to control the motor and receive signals from the rotary encoder. This board is equipped with *Xbee* for wireless communication and a Micro SD card for auxiliary memory. The power for the motor and the board is provided by a rechargeable battery for each. The motor, board, and battery are all installed on the first link to enable giant-swing rotations without twining cables around the bar. Considering the weight balance between the first and second links, two steel weights of 100 g each are attached to the end of the second link. As the robot can be programmed on the board, it can learn without communicating with a PC. The motor has a battery life of approximately two hours.

The weight and size of the robot are as in Fig. 2a. The length of the first and second links is 0.2 m. The weight of the first link comprised of the frame, battery, servo motor, rotary encoder, and microcomputer board is 0.95 kg. The weight of the second link, which is the sum of the frame and the weights, is 0.53 kg. In accordance with the robot, the parameter settings for the simulator are readjusted from (Uragami et al., 2014). The simulator uses a free physical engine, namely the Open Dynamics Engine (ODE). The weight and size of the simulated robot is shown in Fig. 2b. The length of the two links are the same as that of the real robot, 0.2 m. For the weight of the first link, we distributed the 0.95 kg uniformly over the length of 0.23 m between the first and second joints. Similarly, the weight of the second link is based on the uniformly distributed 0.5 kg over the 0.12 m length from the tip of the robot. The weight distribution is adjusted to reproduce the mechanical properties of the real robot.

Fig. 3a shows the comparison between the real and simulated robots in terms of the mechanistic property of the first joint and

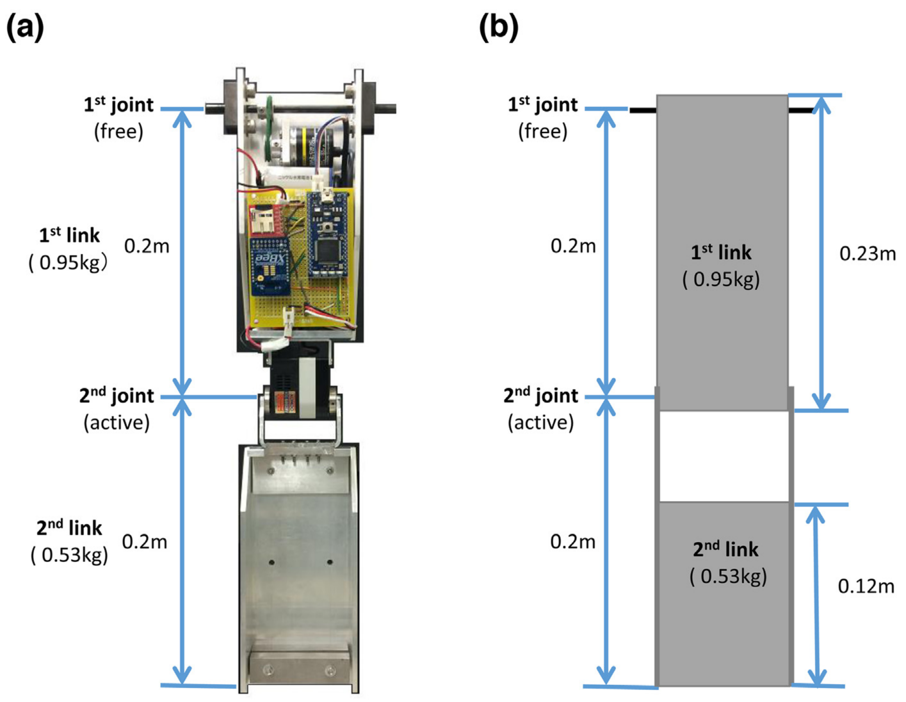


Fig. 2. Dimensions and weight of the (a) robot and (b) simulated robot. The length of the links, total weight of the simulator, and weight balance of the simulator are adjusted in accordance with the real robot.

the motor characteristics of the second joint. The rotation angle is defined as in Fig. 3b. The second joint is set to move from 0 to 0.79π rad. The properties of the motor in the simulator are coordinated with the real one: the torque in the simulator is set to 4.7 rad/s when bent, 5.0 rad/s when stretched, and a maximum 2.40 Nm; a lag of 0.01 s is set when the action is switched. Both the real and simulated robots begin the episode with a straight posture and in a stationary state with the angle of the first and second joints as 1π rad and 0 rad, respectively. After 1 s, they both start to bend the second joint, and 0.6 s later, reaches 0.79π rad. The average angular velocity is approximately 1.3π rad/s. The posture is maintained until 3 s, at which time the second joint starts to stretch, and 0.6 s later, it reaches a minimum of 0π rad in the range of motion. The rotation angles of the first and second joints of the robot and the simulation coincide, for the duration of the experiment (5 s), as shown in Fig. 3a.

2.2. Application of the LS-Q model

We describe how the robot learns in this subsection. The learning setting and environment is as per (Uragami et al., 2014). The states, actions, and rewards are defined in Fig. 4. The state is defined as the product of position, velocity, and posture. The position is defined as the angle of the first joint, uniformly divided into twelve discrete states, from P0 to P11 (Fig. 4a). The velocity is defined as the angular velocity of the first joint uniformly divided into seven states, from W0 to W6, as in Fig. 4b. The posture is the angle of the second joint uniformly divided into five states, from R0 to R4. The total number of states, which is the product of the number of positions, velocities, and postures, is $12 \times 7 \times 5 = 420$. There are three actions: the second joint halted (A0), bent (A1), and stretched (A2). The reward r is defined as follows:

$$r = |\theta_{\mathrm{tip}}/\pi|. \tag{1}$$

where θ_{tip} tip is the angle of the tip of the second link and the horizontal bar. The reward is a minimum, $r = 0$, when the tip is directly below the bar, and is a maximum when the tip is directly above the bar, $r = 1$. The setting is the same as in (Uragami et al., 2014) and is based on the consideration in (Toyoda et al., 2010).

We adopted the LS-Q algorithm as the learning mechanism, and the normal Q-learning algorithm for comparison. Q-learning pro-

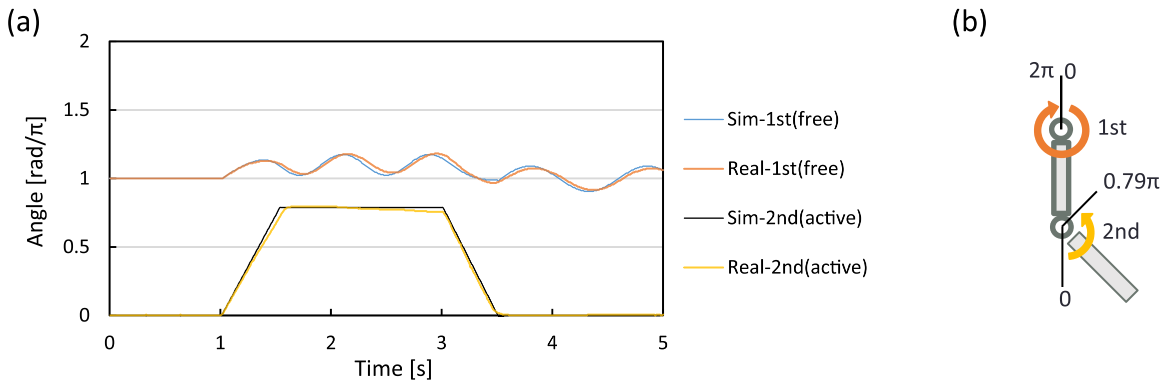


Fig. 3. (a) Comparison of the dynamics of the robot and the simulation: the rotation at the first joint and the characteristics of the servomotor at the second joint coincide. (b) Definition of the rotation angle for the first and second joints.

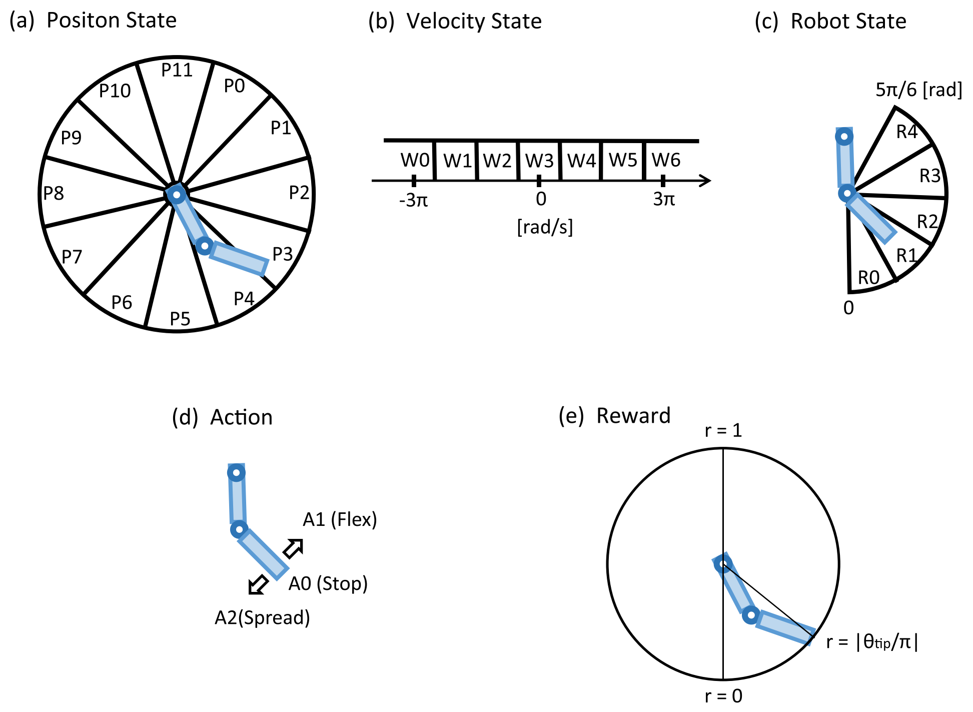


Fig. 4. Definition of state, action, and reward. The state is defined as a triplet (position, velocity, posture). The three variables are equally divided. (a) Position is the rotation angle of the first joint for twenty states, (b) velocity is the rotation velocity of the first joint for seven states, and (c) posture is the movable scope of the second joint for five states. Hence, the total number of states is $12 \times 7 \times 5 = 420$. (d) There are three actions: halting, bending, and stretching the second joint. (e) Reward is given proportional to the angle between the tip of the second link and the bar. When the tip directly below the bar, the reward is 0.0 (the minimum), and when it is directly above the bar, the reward is 1.0 (the maximum).

ceeds by updating the evaluation of action a at state s (Q-value for the state-action pair (s, a)) by the following equation:

$$Q(s,a) \longleftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q\left(s', a'\right)\right], \qquad (2)$$

where α and γ are the learning and discount rates, respectively. Based on the simulation result in (Uragami et al., 2014), we set $\alpha = 0.9$ and $\gamma = 0.9$. r is the reward acquired at (s, a) defined by Eq. (1). s' is the state transited from s by taking action a. $\max_{a'} Q(s', a')$

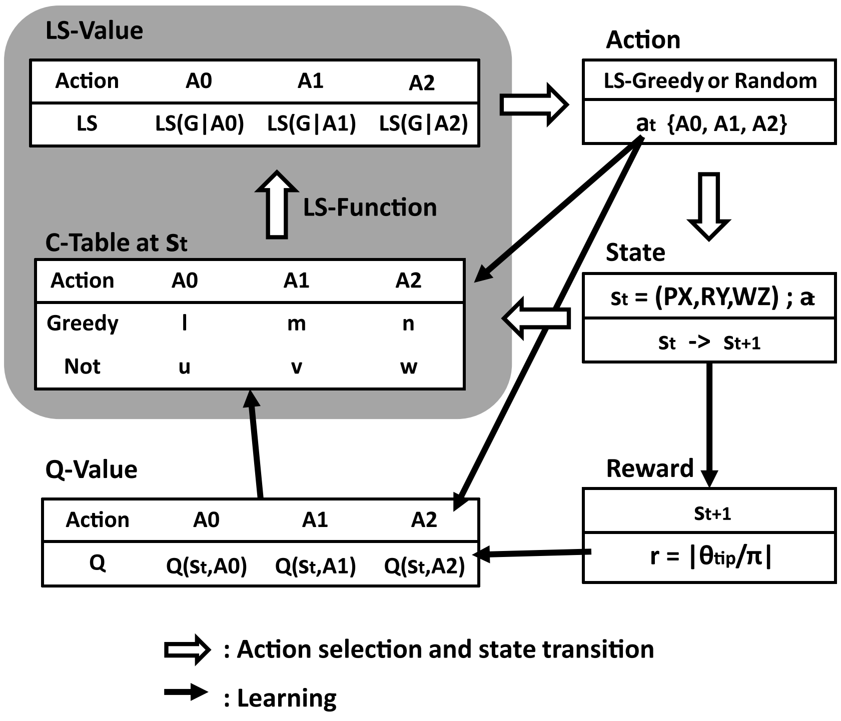


Fig. 5. Scheme for action selection and learning in LS-Q. Action selection is based on the calculation of the LS-values in C-Table. Learning proceeds through updating of the Q-values and C-Tables. The C-Table for each state holds the co-occurrence frequencies between action selection and whether the action was Q-greedy. G in the figure indicates greedy.

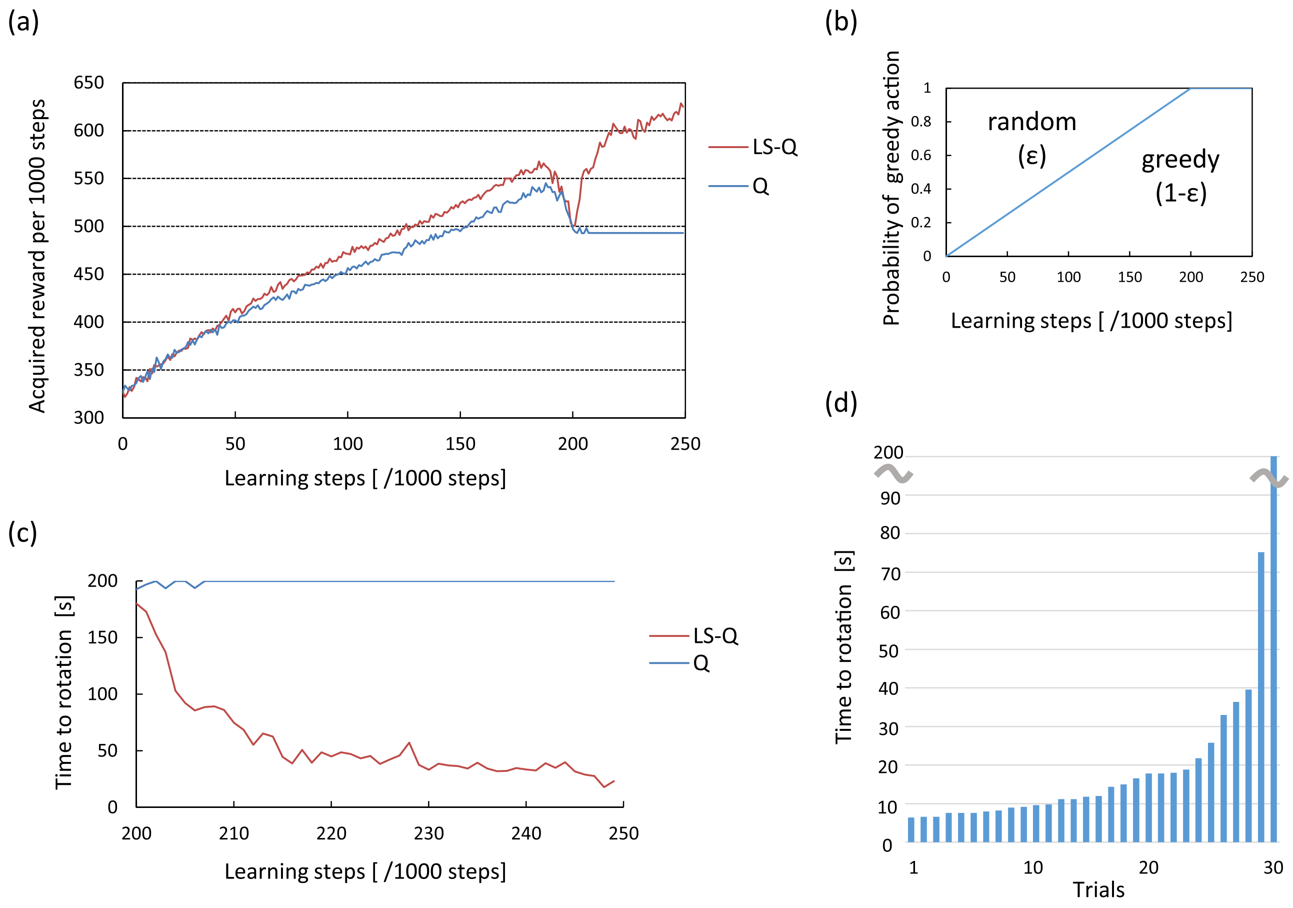


Fig. 6. (a) Learning curves of LS-Q and Q, averaged over 30 simulations. (b) Development of the probability of selecting the (LS-)greedy action (ε). At the beginning of learning, the action is totally randomly chosen. Then, gradually the probability of choosing the (LS-)greedy action increases, and after phase 200 the (LS-)greedy action is chosen. (c) Time taken until rotation, after phase 200, averaged over 30 simulations. (d) Time until rotation for the 30 simulations at the last phase, 250, sorted in ascending order.

is the Q-value in case action a' that maximizes the value at s'. The action with the maximum Q-value at a certain state is called the greedy action at that state.

$$\text{A is the greedy action at S} \quad :\Leftrightarrow Q(S, A) = \max_{a'} Q(S, a') \tag{3}$$

Hence, $\max_{a'} Q(s', a')$ is the Q-value of the greedy action at state s'. In the learning environment of this study, state s and action a are defined as follows:

$$s \in \left\{P0, \ldots, P11\right\} \times \left\{W0, \ldots, W6\right\} \times \left\{R0, \ldots R4\right\}, \tag{4}$$

$$a \in \left\{A0, A1, A2\right\}. \tag{5}$$

$Q(s, a)$ is defined for (420 states(4) $\times$ 3 actions). The action selection follows a policy of mapping from states to an action, which may be probabilistic. By iterating action selection, state transition, and reward acquisition, the updated Q-value, $Q(s, a)$, is gradually determined.

A schematic illustration of the LS-Q algorithm is shown in Fig. 5. The LS-Q model is different from Q-learning in that it has an additional memory called C-Table, and the actions are evaluated using the LS model value defined by the C-Table (gray region in Fig. 5). The C-Table records the history of action selection at a state and is a function $C(s, a, g)$, where g is a variable that represents whether action a is greedy or not, i.e., if it has the maximum Q-value at state s compared to other actions:

$$g \in \{\text{greedy, non-greedy}\}. \tag{6}$$

At time step t, if the agent is at state s_t and chooses action a_t, whether a_t is the greedy one or not (g_t), the C-value is updated as follows:

$$C(s_t, a_t, g_t) \longleftarrow C(s_t, a_t, g_t) + 1 \tag{7}$$

The C-Table stores the frequency of actions being greedy (or non-greedy) at a certain state. The values in the C-Table for state s_t are represented by l, m, n, u, v, and w in Fig. 5. For instance, if $a_t = A0$ and g_t = greedy, then l is incremented by one, while m to w stay constant. From the C-Table, the LS values are computed as follows:

$$\text{LS}(greedy|A0) = \frac{l + u \times (v+w)/(u+v+w)}{l + u \times (v+w)/(u+v+w) + u + l \times (m+n)/(l+m+n)} \tag{8}$$

$$\text{LS}(greedy|A1) = \frac{m + v \times (u+w)/(v+u+w)}{m + v \times (u+w)/(v+u+w) + v + m \times (l+n)/(m+l+n)} \tag{9}$$

$$\text{LS}(greedy|A2) = \frac{n + w \times (u+v)/(w+u+v)}{n + w \times (u+v)/(w+u+v)\} + w + n \times (l+m)/(n+l+m)} \tag{10}$$

LS$(greedy|A)$ is the conditional probability-like value of action A, defined as the probability that A is greedy with respect to the Q-value. The original LS model was defined as a subjective estimation of conditional probability, in particular when considering the

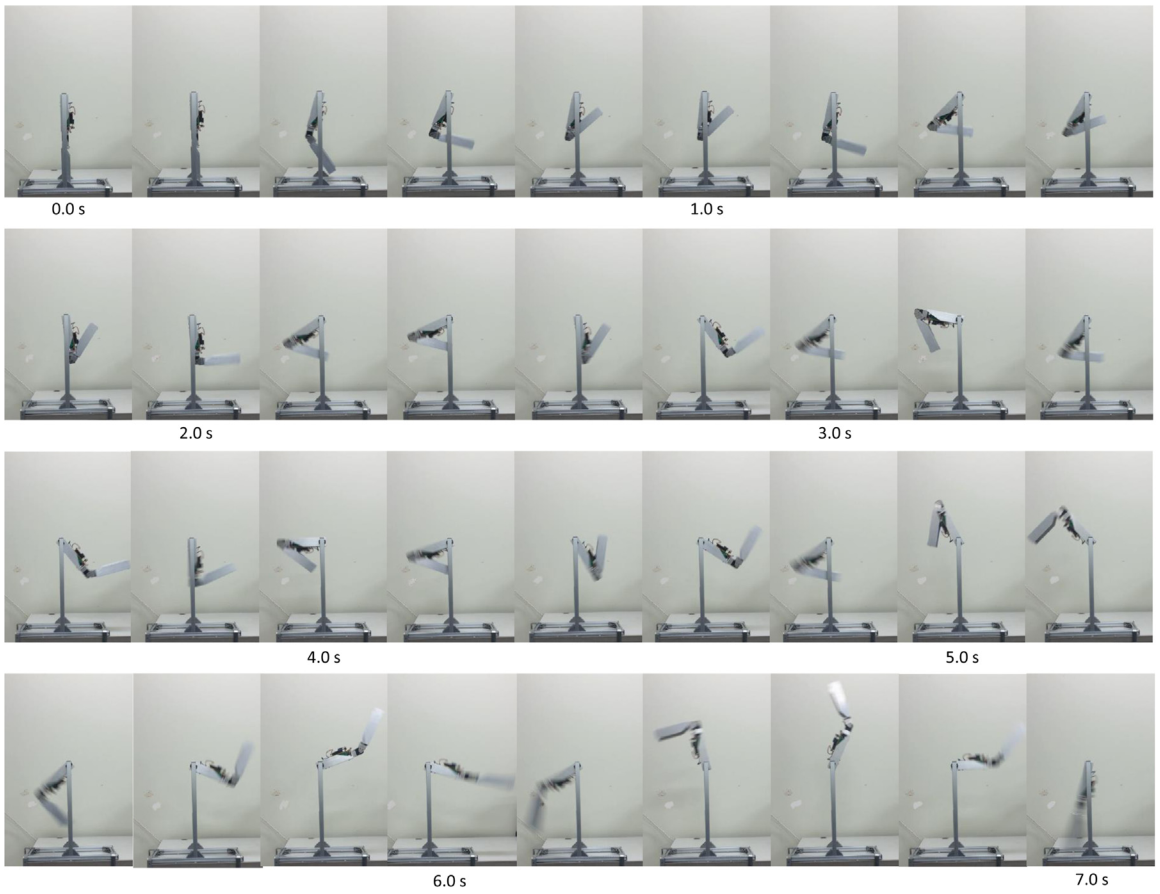


Fig. 7. Rotation of the robot required by the LS-Q, shown in continuous shots at 0.2 s intervals, arranged from the upper left. The robot executed the action sequence transplanted to it from the simulation. By bending and stretching the second joint timeously, the robot amplified the swings at the first joint, and finally rotated clockwise (backward).

causal relationship between events (Shinohara et al., 2007). The LS value has a very high correlation with the experimental results of causal induction by humans, and there is an interpretation of the definition employing the framework of empirical Bayes (Oyo et al., 2015).

In the LS-Q learning algorithm, we define action A as LS-greedy when

$$\text{A is greedy action at S} :\Leftrightarrow LS(\text{greedy}|A) = \max_{a'} \text{LS}(\text{greedy}|a') \qquad (11)$$

During the learning process, the action (LS-greedy or otherwise) is chosen, according to the action selection policy. The learning proceeds through action selection, state transition, reward acquisition, and updating of the Q-value and C-Table, as shown in Fig. 5. In this study, one step in the learning process is defined as action selection, and learning is executed every 0.2 s. We train the simulator to readjust the parameters according to the robot, and record the action sequence and action value function (LS-value and Q-value). Then, we use the action sequence and the values for the robot in the experiments.

3. Results

3.1. Simulation results

Fig. 6 shows the learning simulation result. We define 1000 steps as a phase, and there is a total of 250 simulation phases. In the initial state, the robot remains stationary directly under the bar with the second joint expanded into a straight position. At the beginning of each set, the state is reset to the initial state. The action selection policy is ε-greedy, where the agent selects the action to be random at probability ε. As shown in Fig. 6b, the robot randomly selects the action at the beginning of the learning process. Then, up to phase 200, ε decreases linearly, decremented by 0.005 in each phase. After phase 201, $\varepsilon = 0$ and the greedy (LS-greedy for LS-Q) action is always chosen. The initial value of each Q is 0.

We show the learning curve in Fig. 6a. The x-axis is the learning time (phase), and the y-axis the acquired reward. The reward is the total for the 1 phase = 1000 steps. The thick red and thin blue line shows the result of LS-Q and Q-learning, respectively, averaged over 30 simulations. We see that from phase 50–180, the reward for LS-Q gradually increases above Q. At a later stage, we will analyze the effect of the stagnant loops, which occurs from approximately phase 180, where the reward for both suddenly decreases. After phase 200, the reward for Q remains lower, while for LS-Q it recovers and reaches approximately 600. In summary, faster learning and avoidance of stagnant loops are the characteristics of LS-Q, compared with Q. These are the findings reported in (Uragami et al., 2014), and confirmed in this study by the simulator readjusted according to the robot.

We show the time taken for first rotation of the robot in each learning phase in Fig. 6c. In the plot, the x-axis is learning time, and the y-axis the time until the first rotation, after phase 200, averaged over 30 simulations. One phase runs for 200 s (1000 steps $\times$ 0.2 s). If the agent does not rotate in the phase, then the rotation time is defined as 200 s. We see that by Q-learning no rotation occurs after approximately phase 210. The LS-Q enables shorter time until rotation as the learning proceeds, and the time to rotation reaches approximately 25 s. Fig. 6d shows the time to rotation at the final phase, phase 250, for all 30 simulations for LS-Q. There are 12 trials with the time to rotation within 10 s, 24 trials in total within 20 s. Only one simulation did not reach rotation within 200 s. However, we found that this simulation finally reaches rotation at phase 317, and after phase 319, it starts to rotate within 11.4 s.

3.2. *Robot results*

We test the validity of the simulation result with a robot. First, we use the sequence recorded in the simulation at phase 250 as the action selection sequence of the robot. Fig. 7 shows the stop-motion images taken every 0.2 s. We can see that the robot increases the swing width at the first joint, and the second joint is rhythmically actuated. At 6.6 s we see the robot reaching directly above the bar, leading to rotation. Fig. 8a shows the comparison of the LS-Q robot and its simulator in terms of the rotation angle of the first and second joints. We find that the robot and the simulator show nearly the same trajectory of rotation. Similarly, Fig. 8b shows the result for Q. For both the robot and simulation, we see that the second joint is fixed swung up, and the first joint narrowly swings around directly below the bar, which does not lead to rotation.

Next, we used the action values (LS and Q values) learned in the simulation. Fig. 8c shows the plot of the rotation angle of the first and second joints of the robot. Comparing the LS-Q in Fig. 8a and c, we see that the movement of the second joint differs slightly between the robot and simulation, and the trajectories of the first joint are different. This is due to the difference in state transition and observation resolution in the robot and simulator. However, for LS-Q in Fig. 8c, although it took longer than in Fig. 8a, the robot begins to rotate at approximately 14 s. On the other hand, for Q-learning, the robot did not realize rotation, as shown in Fig. 8c, where, similarly to Fig. 8b, the second joint is fixed as bent. In Fig. 8d, we show the time to rotation for ten experiments, where the LS values for the LS-Q simulations are shown to have a quicker time to rotation. For all ten trials, we find that the robot rotates in the range from 10 s to 20 s. A similar experiment for Q showed no rotation in all ten trials.

4. Analysis and discussion

We confirmed the validity of the LS-Q algorithm for real robots by the experiments, in the previous section. Here, we analyze the conditions for stagnant loops and the mechanism by which the LS-Q escapes the loops. Our discussion here is based on the simulation result that reproduces the characteristics of the robots.

As shown in Fig. 8b, the actions acquired by Q-learning go into the loop of the second joint maintained in the bent position, with the second link swung up. Sakai and others discovered this phenomenon and termed it “stagnant loops” (Toyoda et al., 2010). Because the reward is given according to the angle of the position of the tip of the second link, stretching the second joint from the swung up to the straight position temporarily decreases the reward. However, this is an action that is necessary for the robot to obtain a higher position around the bar, and consequently acquire more reward. In principle, Q-learning can acquire an action sequence with more reward, given that the discount rate is appropriately set. However, as in the environment in this study, when the continuous state space is divided into coarse-grained discrete states, Q-learning often fails to learn appropriate action sequences. According to the results in (Uragami et al., 2014) and (Toyoda et al., 2010), the cause of the stagnant loops is non-deterministic state transition and/or non-Markov property resulting from incomplete state observation. Therefore, we analyze the detailed state transition inside the stagnant loops and elucidate the relationship between the non-deterministic state transition

Table 1
States and actions taken during the initial 25 steps in phase 201 and 210.

Steps	State (201)	Action (201)	State (210)	Action (210)
0	190	A1	190	A1
1	226	A1	226	A1
2	228	A1	228	A1
3	189	A0	189	A0
4	194	A1	194	A2
5	234	A1	227	A1
6	229	A1	234	A1
7	189	A0	224	A1
8	194	A1	194	A2
9	229	A0	192	A1
10	229	A0	269	A1
11	194	A1	259	A1
12	194	A1	184	A2
13	229	A1	157	A1
14	229	A1	279	A1
15	194	A1	334	A0
16	194	A1	214	A2
17	229	A0	117	A1
18	229	A0	209	A1
19	194	A1	344	A1
20	199	A1	249	A1
21	229	A1	109	A2
22	224	A1	127	A1
23	194	A1	244	A1
24	199	A1	379	A2
25	229	A1	403	A0

and the stagnant loops. Then, we observe how the LS-Q escapes from the loops.

We analyze a typical simulation result from the 30 simulations shown in Fig. 6. Fig. 9a shows the time to rotation from phase 201–210. These are the first ten phases in the simulation where the action is totally chosen deterministically. As described in the previous section, one phase consists of 1000 steps (200 s), and when there is no rotation, the time to rotation is defined as 200 s. As in Fig. 9a, no rotation is realized until phase 204 and from phase 205 rotation is realized at 6.6 s (33 steps). Fig. 9b shows the trajectory of the robot in phase 201 and 210 from 0 to 5 s. We find that the cause of the lack of rotation is the stagnant loop in which the robot halts with the second joint swung up. When the robot rotates, the loop is avoided.

To clarify the difference between the cases where the stagnant loop occurs and where rotation occurs, we show all the action selection and state transition in the first 25 steps (5 s) in phase 201 and 210 in Table 1. We present the 420 states (combination of the twelve positions, seven velocities, and five postures as shown in Fig. 4), by a number from 0 to 419, defined as:

$$(Pi, Wj, Rk)\,\text{is state}(35i + 5j + k) \tag{12}$$

For example, the initial state is state 190, $(P5, W3, R0)$. We find state 194, $(P5, W3, R4)$ several times in Table 1. In phase 201, action $A1$ is chosen every time at state 194. This state-action pair leads to several transitions to states such as 231, 229, 194, 199. At phase 210, at state 194, action $A2$ is chosen twice, and the next states (227 and 194) are multiples. Namely, at state 194, the current state and action does not determine the next state. Fig. 9c shows the state transition generated from Table 1. The numbers in the circles are state numbers. The solid arrows and the numbers on the side represent state transitions and the actions chosen at phase 210, respectively. In phase 210, the robot reaches rotation. Starting from state 190, action A1 (bending the second joint) is chosen. After passing states 226 and 228, A0 (halting the second joint) is chosen at state 189 and reaches state 194. As in Fig. 9c, there are multiple transitions from state 194 with action A2, to state 227 and 192. When it reaches 227, it reverts to 194 through 234 and 224. After it reaches 192, through a linear transition sequence, it leads to 403.

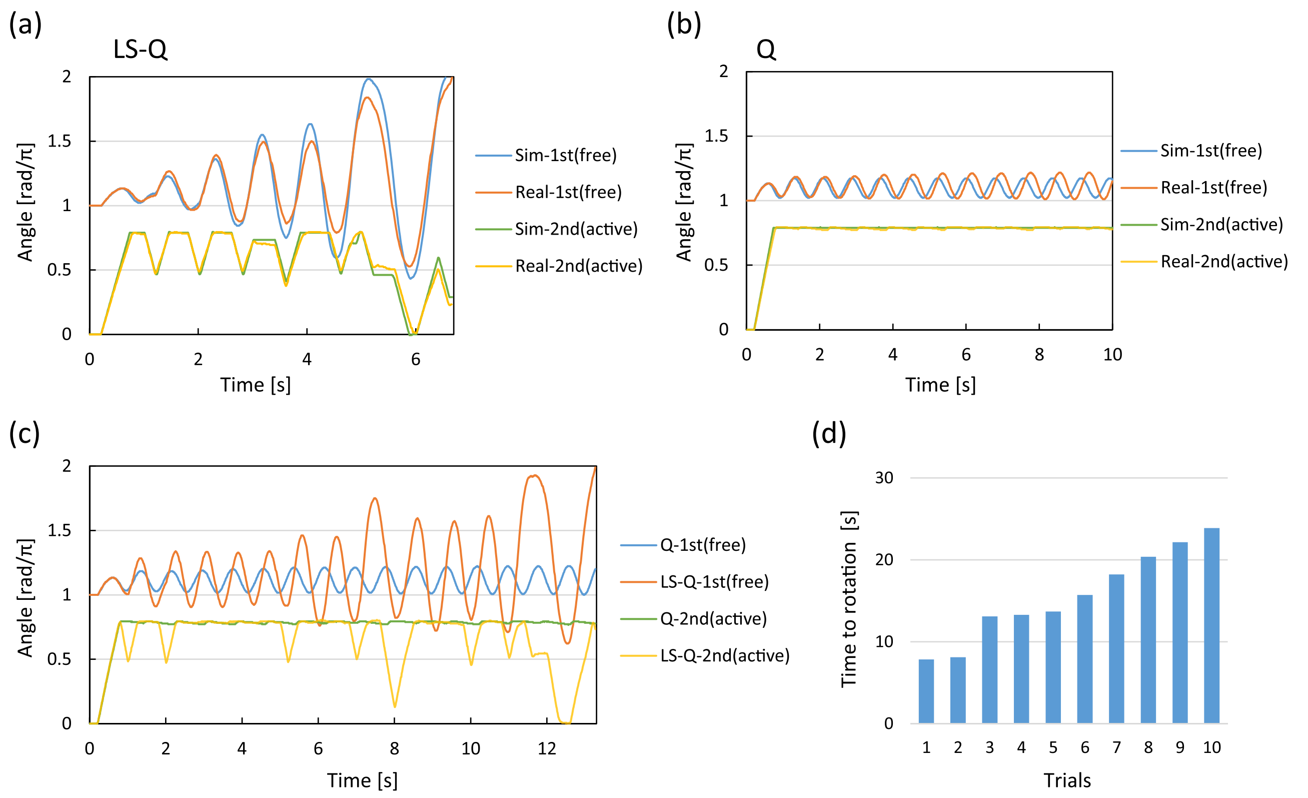
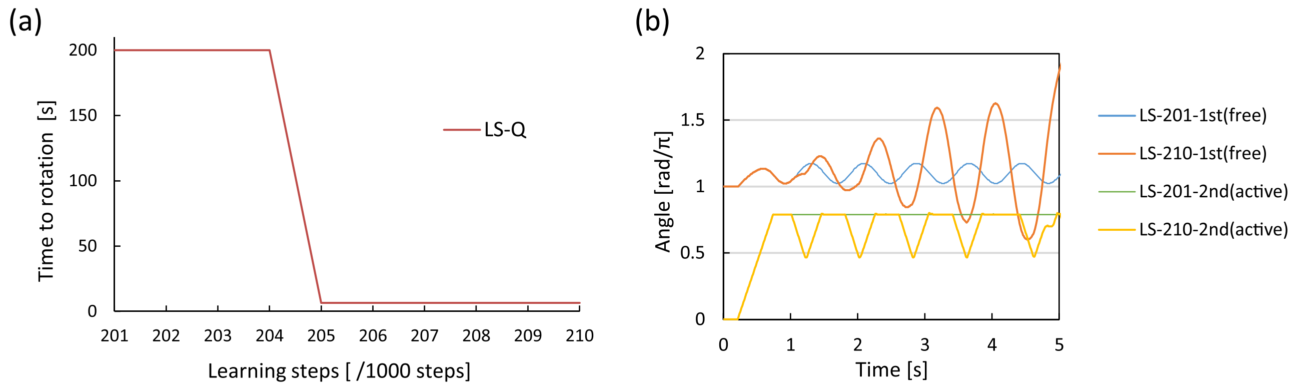


Fig. 8. Trajectories of the first and second joints in the movement acquired by the LS-Q, in the robot (Real) and simulation (Sim). (b) Trajectories by Q-learning. (c) Trajectories by LS- and Q-values, transplanted from the simulator to the robot. (d) Sorted times until rotation for the ten robot trials where the LS-values are transplanted from the simulations.



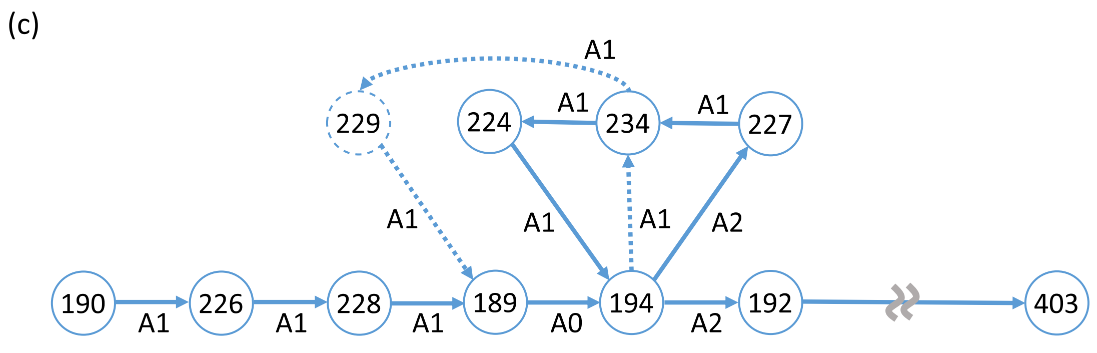


Fig. 9. Development of the time until rotation, in a typical trial, from phase 201 and 210, where the LS-greedy action is always chosen. (b) Comparison of the movement of the robot in phase 201 and 210, in the same trial as in (a). (c) Part of state transition in Table 1, from the initial state (190) to rotation (403) in phase 210, where the number in the node circle is state number and arrow label is action chosen. The solid arrows and nodes are the actions and state transited. Dotted arrows and nodes are part of the stagnant loop in phase 201.

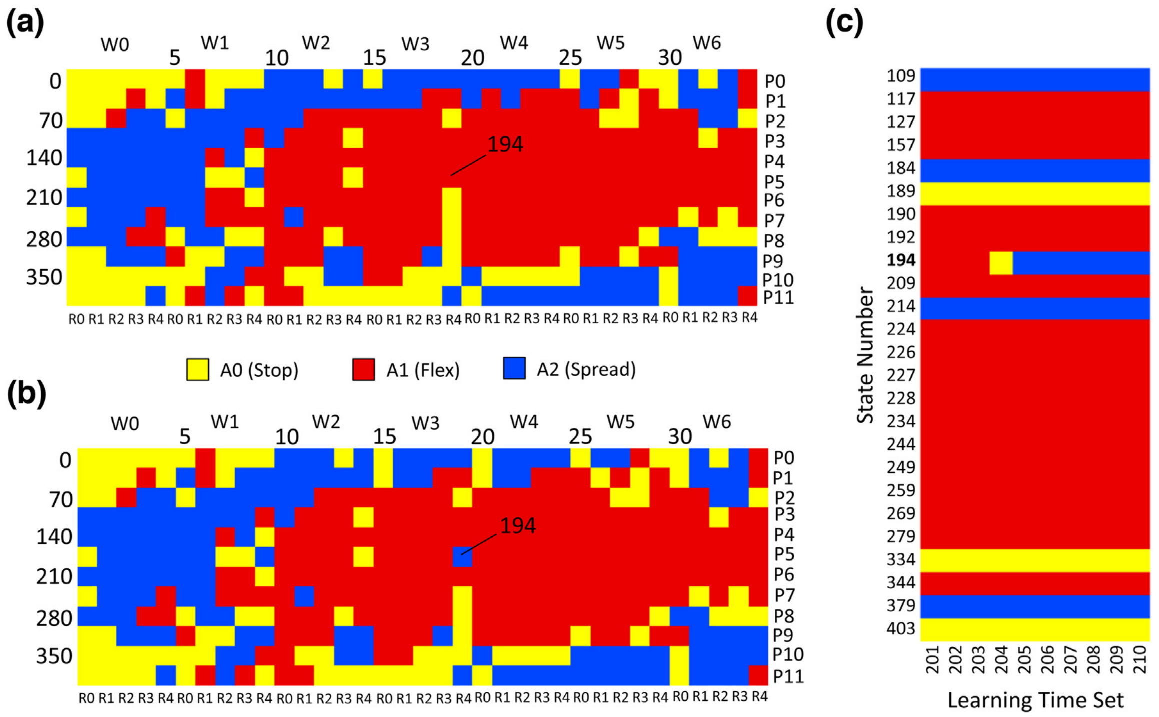


Fig. 10. (a) A policy of LS-Q acquired by learning, which is the table of actions that will be taken at each state, at phase 201. The three actions are colored as yellow A0 (halt), red A1 (bend), and blue A2 (stretch). (b) A policy of LS-Q at phase 210. (c) Development of the policy at 25 states that are passed through from the initial state to rotation, from phase 201–210. We see that it is only at state 194 that the LS-greedy action alters. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The transitions from 194 are considerably more complicated when we consider the case where action A1 is chosen (shown with dashed arrows). Such a non-deterministic state transition results from an incomplete state observation (partially observable environment).

Next, to see how the agent learns to act, we compare the action value functions in both cases. Fig. 10a and b show the LS-greedy actions for each state by three colors, yellow (A0, hold), red (A1, flex), and blue (A2, spread) at phase 201 and 210, respectively. There are 12 rows and 35 (5×7) columns, in total 420 cells, that correspond to the possible states. These two figures show the greedy policy (a mapping from states to an action). There are some states where the LS-greedy action differs between (a) phase 201, and (b) 210, as the learning proceeds. One of the most prominent changes is at state 194, where in (a) its LS-greedy action is red (A1) as for the other states in red, whereas in (b) it changes to blue (A2). Fig. 10c shows the time development of the LS-greedy actions for the 25 states that are passed prior to rotation. As for the states relevant to rotation, it is only state 194 that changes the LS-greedy action. From Figs. 9 c and 10 c, we find that it is at phase 205 that the agent acquires a policy that leads to rotation, and it is then that the LS-greedy action at state 194 changes from A1 via A0–A2. In summary, we find that state 194 is a state where the non-deterministic state transition pronouncedly appears, and the action taken at this state is the factor that divides the cases according to whether the agent can lead to rotation or not.

Then, we analyze the mechanism of falling into a stagnant loop when the action is chosen greedily, and the method by which LS-Q escapes from it. Fig. 11a shows the plot of the time development of the values of the C-Table ($l, \ldots, w$) for state 194 from phase 201–210 (10,000 steps). This development can be compared with the LS-values (defined for the three actions as in (8–10)) calculated from the values in the C-Table, as in Fig. 11b, and the Q-values that are used for the definition of the C-Table, as in Fig. 12a. In Fig. 11a, first the value of m increases because, as in Fig. 11b, A1 is the LS-greedy action and, as in Fig. 12a, A1 is the (Q-) greedy action. Then, from approximately step 200,450, it is v that is incremented, because A1 ceases to be the greedy action, as in Fig. 12a. As v increases, the LS-value of A1 ($LS(\text{greedy}|A1)$) decreases. From approximately step 201,500, the precedence of the LS- and Q-values of A0 and A1 frequently switches, and the values l, m, u, v increase. In the long term the LS- and Q-values for A0 and A1 decrease. From approximately step 204,000, the LS- and Q-value for A2 is relatively larger than for A0 and A1. This is the learning process where the LS-greedy action at state 194 changes from A1 to A2.

Summarizing the above analyses, we see that the two causes that enable action A2 are finally chosen at state 194. One is the property of the LS-values to change and consequently switch among more values than the non-biased conditional probability $P : P(\text{greedy}|A0)$, $P(\text{greedy}|A1)$, and $P(\text{greedy}|A2)$. In fact, the performance using P-values instead of the LS-values in the C-Tables is tested in (Uragami et al., 2011), and results were found to lie between those of Q and LS-Q. The difference between LS- and P-values is clear when we compare Fig. 11b and c. As shown in Fig. 11c, P-values fluctuate as the values in the C-Table change, but the order does not change, hence leading to no change in the action selection. The LS-values in Fig. 11b show that $LS(\text{greedy}|A2)$ increases, even though up until approximately step 204,000, A2 is not chosen, so n and w do not change. It is because of the relative evaluation resulting from the mutual exclusivity bias (A0 being greedy when chosen is equivalent to other actions A1 or A2 being non-greedy when chosen) that the increase in choice of the other actions, A0 and A1, when non-greedy, leads to the increase in $LS(\text{greedy}|A2)$. As a result, A2 becomes the LS-greedy action from approximately step 204,000.

The other cause is that the action with the highest Q-value tends to alter more with LS-Q than Q. This process of alteration is shown in Fig. 12a where the Q-values are plotted in the same episode and states as in Fig. 11, from phase 201–210, for 10,000 steps. The

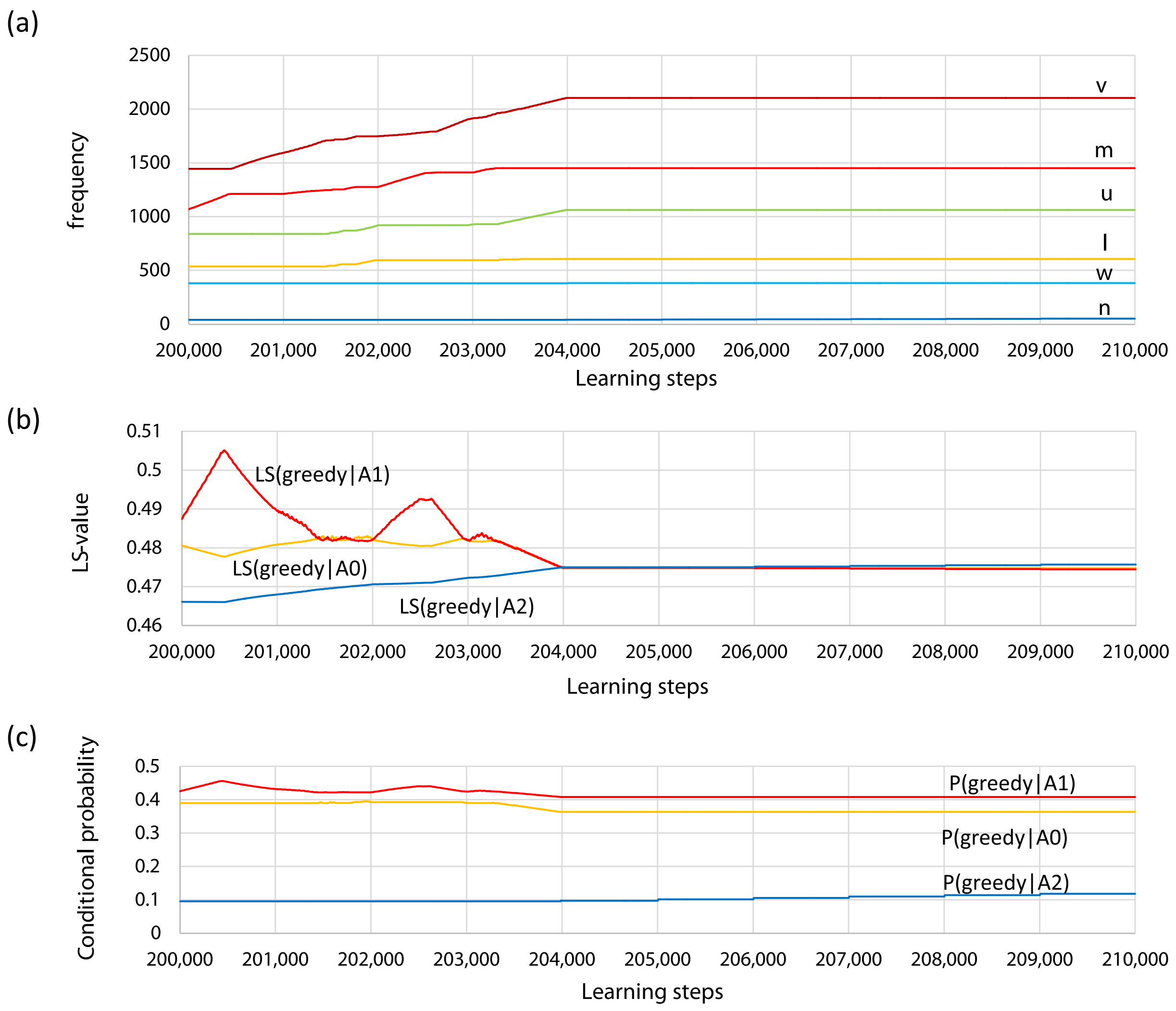


Fig. 11. (a) Development of the values on C-Table at state 194, from phase 201–210. (b) Development of the LS-values at state 194 that are determined from (a). (c) Development of the P-values (unbiased conditional probability) at state 194 that are determined from (a).

Q-values for A0 and A1 decrease, while fluctuating, to lower than that of A2. After step 5000, A2 maintains the highest Q-value. For comparison, we show the Q-values in Q-learning where the simulated robot cannot escape from a stagnant loop in Fig. 12b. We find that while the Q-value for A0 fluctuates because it is continuously chosen, A0 remains the greedy action. Comparing Fig. 12 (a) and (b), we find that the Q-values for A2 are higher for LS-Q (between 5.3 and 5.4) and lower for Q (around 4.8). On the other hand, the minimum Q-value for A0 and A1 in a stagnant loop is approximately 4.85, as shown in Fig. 12b. If the Q-value for A2 is higher than the value at the beginning of phase 201, this alteration could occur.

As stated above, the Q-values at the end of phase 200 affect the following learning stage. We show the history of the Q-values from phase 1–250, for 250,000 steps, at state 194 in Fig. 12c. The values are averaged over 30 simulations. We see that in Q-learning, the Q-values decrease after an initial increase for 30,000 steps. The appropriate action sequences are not learned. On the other hand, the Q-values in LS-Q keep increasing until approximately phase 150. This means that LS-Q is effective not only for avoiding a stagnant loop, but also for the middle stage of the learning. This effectiveness can be confirmed in Fig. 6a, which shows the development of the acquired reward. After approximately phase 150, the Q-values start to decrease. This appears to be as a result of stagnant loops.

The LS-Q tends to construct action sequences that lead to higher reward in the middle stage of learning. This is one of the causes of escaping from the stagnant loop. In the middle stage, the Q-values fluctuate and the Q-greedy action at each state alters frequently. With the C-Table used in the LS-Q, holding the concurrent frequencies between action selection and the determination of it being a Q-greedy action, the manner in which the LS-Q handles the LS value function is a restless (non-stationary) bandit problem. It is shown that the LS model is effective in some restless bandit problems (Oyo et al., 2015). Further analysis of the relationship between the altering Q-values and C-Tables and LS-values is required in future.

The state description is based on the uniform, coarse-grained state division set by the designer, not by the robotic agent itself. In that sense, the state description is not primary. Therefore the effectiveness of LS-Q is required to be independent from a specific state definition. It has been shown in our previous work (Uragami et al., 2014) that LS-Q can perform well against varying resolution of state description, without parameter adjustment. On the other hand, the non-Markovian property results from the coarse-grained state description. However, since LS-Q utilizes the C-Table and the LS value function without reconstructing the dynamics of the learning environment, it can be expected that LS-Q performs well in more originally non-Markovian environments. C-Table itself is effective when the state transition can be considered approximately

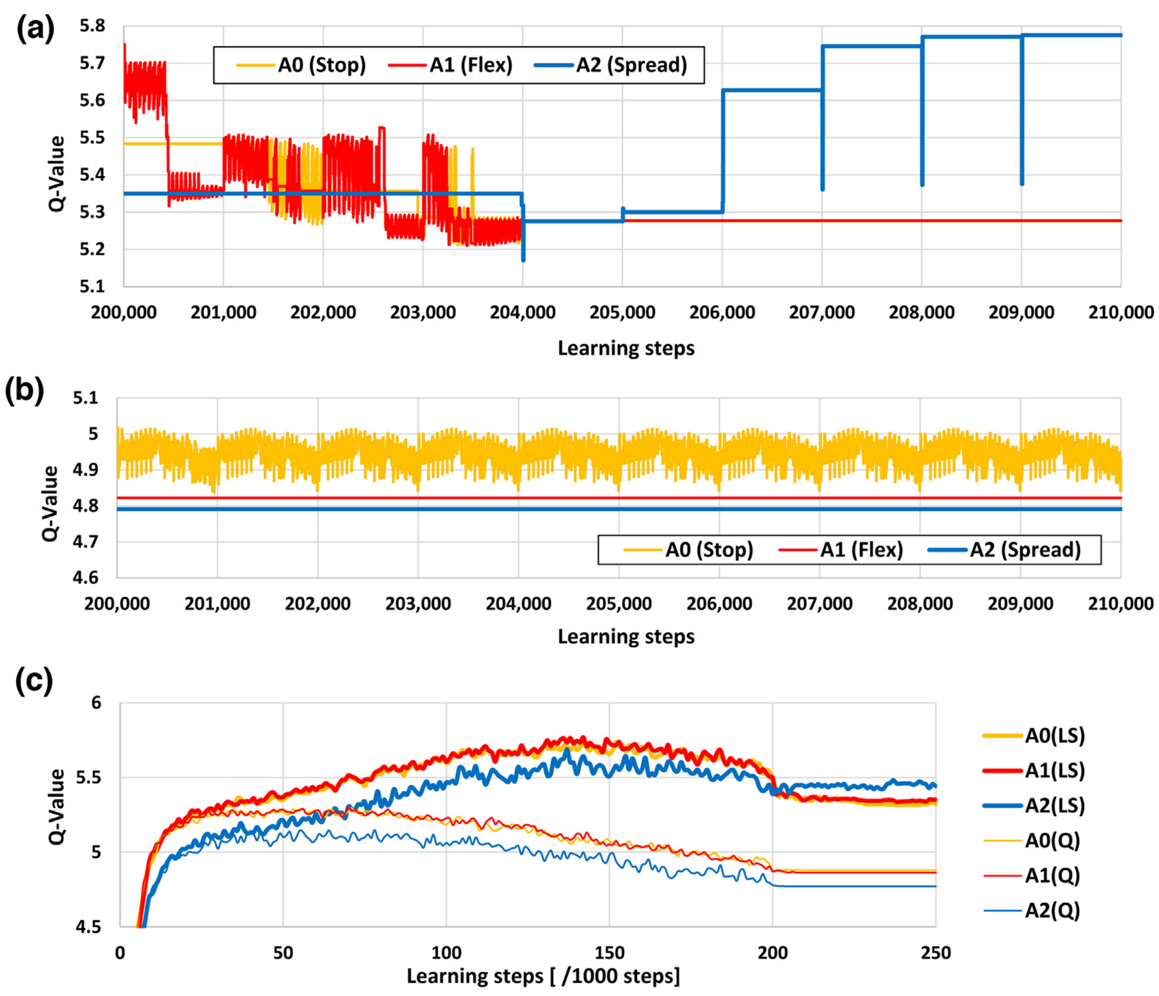


Fig. 12. (a) Development of Q-values at state 194 in LS-Q, from phase 201–210. (b) Development of Q-values at state 194 in Q-learning, from phase 201–210. (c) Development of Q-values at state 194 in LS-Q and Q-learning, for the whole learning time (from phase 1–250).

probabilistic (as shown for the "NS" algorithm in Uragami et al., 2011). The results above showed that the effect of adopting the LS value function becomes more prominent when stagnant loops occur. Because state 194 corresponds to a set of physical states, choosing the same action at the state leads to transiting to multiple states. However, choosing action A2 twice in a row at state 194 lets the robot escape from the stagnant loop. In the learning environment in this study, while the non-Markovian property makes the learning harder in the learning process, in the result of learning, it is enough to learn to choose an action at each state. When the robot needs to choose between two or more actions at a state, some mechanism different from C-Table would be necessary. However, if the learning continues and the LS-greedy action alternates with critical LS values, it would not be impossible. This is one of our future issues.

5. Conclusion

In this study, we tested the learning capacity of the LS-Q reinforcement learning architecture with the giant swing robot. We confirmed that LS-Q adaptively learns in an environment where the usual Q-learning goes into stagnant loops and cannot acquire appropriate actions. To elucidate the mechanism by which LS-Q escapes from stagnant loops, we analyzed the state transition and action sequence acquired by LS-Q and Q. As the result, we found that the stagnant loops are caused by non-deterministic state transition (being non-Markovian) from a certain state that is critical for rotation. We further found that LS-Q escapes from a stagnant loop with rapidly changing LS-values. Usually, for the partially observable Markov decision process, such as the learning environment in this study, the dynamics of the environment are reconstructed as the internal model for the agent. LS-Q shows similar results to the more technical methods by imitating and applying human cognitive biases implemented in the LS function. Below, we discuss the meaning, expansivity, and issues of LS-Q.

In general, the states in physical environments that robots sense and handle are continuous or enormous. One way for applying reinforcement learning on such state space is to divide it into coarse-grained states. There is a tradeoff or dilemma in the state division, as the higher resolution leads to a larger number of states and hence more trials are required, whereas, with lower resolution the environment becomes more partially observable (POMDP) and the guarantee for choosing the appropriate action becomes weaker. This usually leads to the requirement for ingenious discretization or differential resolution for state, e.g., applying the higher resolution only to the crucial subspace of the state space, as in (Sutton, 1996). In this case, the method of designing the discretizing resolutions incurred on the continuous state space depends on the decision of the designer. On the other hand, Morimoto and Doya made the learning hierarchical (Morimoto and Doya, 2001), whereas Hara and others devised methods to reward and recognize states (Toyoda et al., 2010). All of the discretizing, hierarchical and reward adjustment methods depend on the theory, previous knowledge, and trial-and-error of the engineers/researchers who design the learning scheme.

If we deal with an enormous state space with reinforcement learning as it is, one way is to use a function approximator. The DQN is a successful example of the use of a function approximator realized by deep neural networks (Mnih et al., 2015b). Its basic idea is to realize a nonlinear approximation function for Q-values with sensor information and the time-difference (TD) error. The DQN learns the nonlinear approximation function for Q-values with the input of sensor information and reward. The approximation function has a distributed representation, and this leads to difficulty in explaining how each state in classical reinforcement learning is represented. Thus far, the method of executing the learning of state recognition has not been argued. On the other hand, animals and humans seem to learn how to recognize and define states and learning action values simultaneously. The symbolization may be a prerequisite for fast transfer learning and higher order action planning. In this study, we used a robot and simulations to show that LS-Q can learn appropriate action sequences under uniformly coarse-grained, hence incomplete, state recognition, without any prior knowledge of the environment. Autonomous symbolization of state recognition is not within the scope of this study. Ideally, the method by which an agent recognizes states should be learned based solely on reward. In this case, it is important to keep learning in the initial stage avoiding stagnant loops is important. When combined with autonomous symbolization of states, the effective learning under incomplete state recognition is crucial because in the initial stage state recognition is vague. In this sense, LS-Q may be shown to be important.

Acknowledgements

We thank Prof. Yoshiki Matsuo and Mr. Ali Alalwan at Tokyo University of Technology who helped build the robot. Part of this work was carried out with the support of the Cooperative Research Project Program H25/A12 of the Research Institute of Electrical Communication, Tohoku University; JSPS KAKENHI 25730150 and 26–10453.

References

Auer, P., Cesa-Bianchi, N., Fischer, P., 2002. Finite-time snalysis of the multiarmed bandit problem. Mach. Learn. 47, 235–256.

Barsalou, L.W., 1999. Perceptual symbol systems. Behav. Brain Sci. 22, 577–609, http://dx.doi.org/10.1017/S0140525X99252144, Discussion 610–660.

Genewein, T., Leibfried, F., Grau-Moya, J., Braun, D.A., 2015. Bounded rationality abstraction, and hierarchical decision-Making: an information-theoretic optimality principle. Front. Robot. AI 2, 27, http://dx.doi.org/10.3389/frobt.2015.00027.

Hauser, J., Murray, R.M., 1990. Nonlinear controllers for non-integrable systems: the acrobot example. 1990. Am. Control Conf., 669–671.

Kaelbling, L., Littman, M., Cassandra, A., 1998. Planning and acting in partially observable stochastic domains. Artif. Intell. 101, 99–134, http://dx.doi.org/10.1016/S0004-3702(98)00023-X.

Kohno, Y., Takahashi, T., 2015. A cognitive satisficing strategy for bandit problems. Int. J. Parallel Emergent Distrib. Syst. 5760, http://dx.doi.org/10.1080/17445760.2015.1075531.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444, http://dx.doi.org/10.1038/nature14539.

Markman, E.M., Wachtel, G.F., 1988. Children's use of mutual exclusivity to constrain the meanings of words. Cognit. Psychol. 20, 121–157, http://dx.doi.org/10.1016/0010-0285(88)90017-5.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015a. Human-level control through deep reinforcement learning. Nature 518, 529–533, http://dx.doi.org/10.1038/nature14236.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015b. Human-level control through deep reinforcement learning. Nature 518, 529–533, http://dx.doi.org/10.1038/nature14236.

Morimoto, J., Doya, K., 2001. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. Rob. Auton. Syst. 36, 37–51, http://dx.doi.org/10.1016/S0921-8890(01)00113-0.

Munos, R., Moore, A., 2002. Variable resolution discretization in optimal control. Mach. Learn. 49, 291–323, http://dx.doi.org/10.1023/A:1017992615625.

Niizato, T., Gunji, Y.-P., 2011. Applying weak equivalence of categories between partial map and pointed set against changing the condition of 2-arms bandit problem. Complexity 16, 10–21, http://dx.doi.org/10.1002/cplx.20331.

Ormoneit, D., Sen, Å., 2002. Kernel-based reinforcement learning. Mach. Learn. 49, 161–178, http://dx.doi.org/10.1023/A:1017928328829.

Oyo, K., Ichino, M., Takahashi, T., 2015. Cognitive validity of a causal value function with loose symmetry and its effectiveness for N-armed bandit problems. Trans. Jpn. Soc. Artif. Intell. 30, 403–416.

Pfeifer, R., Lungarella, M., Iida, F., 2007. Self-Organization embodiment, and biologically inspired robotics. Science 318, 1088–1093, http://dx.doi.org/10.1126/science.1145803 (80-.).

Shinohara, S., Taguchi, R., Katsurada, K., Nitta, T., 2007. A model of belief formation based on causality and application to N-armed bandit problem. Trans. Jpn. Soc. Artif. Intell. 22, 58–68, http://dx.doi.org/10.1527/tjsai.22.58.

Sidman, M., Rauzin, R., Lazar, R., Cunningham, S., Tailby, W., Carrigan, P., 1982. A Search for symmetry in the conditional discriminations of rhesus monkeys, baboons, and children. J. Exp. Anal. Behav. 37, 23–44.

Simon, H.A., 1956. Rational choice and the structure of the environment. Psychol. Rev. 63, 129–138, http://dx.doi.org/10.1037/h0042769.

Spong, M.W., 1995. The swing up control problem for the Acrobot. IEEE Control Syst. Mag. 15, 49–55, http://dx.doi.org/10.1109/37.341864.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, London, England.

Sutton, R.S., 1996. Generalization in reinforcement learning: successful examples using sparse coarse coding. Adv. Neural Inf. Process. Syst. 8, 1038–1044.

Takahashi, T., Nakano, M., Shinohara, S., 2010. Cognitive symmetry: illogical but rational biases. Symmetry Cult. Sci. 21, 1–3.

Takahashi, T., Oyo, K., Shinohara, S., 2011. A loosely symmetric model of cognition. LNCS 5778, 238–245.

Tani, J., 1996. Model-based learning for mobile robot navigation from the dynamical systems perspective. IEEE Trans. Syst. Man Cybern. Part B 26, 421–436, http://dx.doi.org/10.1109/3477.499793.

Taniguchi, T., Nagai, T., Nakamura, T., Iwahashi, N., Ogata, T., Asoh, H., 2016. Symbol emergence in robotics: a survey. Adv. Robot. 30 (11-12), 706–728, http://dx.doi.org/10.1080/01691864.2016.1164622.

Toyoda, N., Yokoyama, T., Sakai, N., Yabuta, T., 2010. Realization and analysis of giant-swing motion using Q-learning. 2010 IEEE/SICE Int. Symp. Syst. Integr. SI Int. 2010-3rd Symp. Syst. Integr. SII 2010, Proc, 365–372, http://dx.doi.org/10.1109/SII.2010.5708353.

Uragami, D., Takahashi, T., Alsubeheen, H., Sekiguchi, A., Matsuo, Y., 2011. The efficacy of symmetric cognitive biases in robotic motion learning. In: Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation, August 7–10, Beijing, China, pp. 410–415.

Uragami, D., Takahashi, T., Matsuo, Y., 2014. Cognitively inspired reinforcement learning architecture and its application to giant-swing motion control. Biosystems 116, 1–9, http://dx.doi.org/10.1016/j.biosystems.2013.11.002.