Reinforcement learning accelerated by using state transition model with robotic applications

著者	Senda Kei, Fujii Shinji, Mano Syusuke
journal or publication title	2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
volume	4
page range	3732-3737
year	2004-09-01
URL	http://hdl.handle.net/2297/1847

Proceedings of 2004 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

Reinforcement Learning Accelerated by Using State Transition Model with Robotic Applications

Kei Senda
Kanazawa Univ.
2-40-20 Kodatsuno, Kanazawa,
Ishikawa 920-8667, Japan
Email: senda.k@t.kanazawa-u.ac.jp

Shinji Fujii
Kanazawa Univ.
2-40-20 Kodatsuno, Kanazawa,
Ishikawa 920-8667, Japan

Syusuke Mano
Kanazawa Univ.
2-40-20 Kodatsuno, Kanazawa,
Ishikawa 920-8667, Japan

Abstract— **This paper discusses a method to accelerate reinforcement learning. Firstly defined is a concept that reduces the state space conserving policy. An algorithm is then given that calculates the optimal cost-to-go and the optimal policy in the reduced space from those in the original space. Using the reduced state space, learning convergence is accelerated. Its usefulness for both DP (dynamic programing) iteration and Q-learning are compared through a maze example. The convergence of the optimal cost-to-go in the original state space needs approximately N or more times as long as that in the reduced state space, where N is a ratio of the state number of the original space to the reduced space. The acceleration effect for Q-learning is more remarkable than that for the DP iteration. The proposed technique is also applied to a robot manipulator working for a peg-in-hole task with geometric constraints. The state space reduction can be considered as a model of the change of observation, i.e., one of cognitive actions. The obtained results explain that the change of observation is reasonable in terms of learning efficiency.**

I. Introduction

Future space projects, e.g., solar power satellites for energy acquisition, will need space robots that work instead of astronauts. Such space robots need to recognize their environments to choose suitable actions and to carry out tasks autonomously. Therefore, we must study and develop intelligent and autonomous robots [1]. For the purpose, this study applies the reinforcement learning [2]. However, it often requires so large number of computations that the method cannot apply to real problems. It is necessary to accelerate the learning speed.

Investigations of skilled human operators point out a change of “observation” [3]. One’s environmental observation changes to indicate efficient and right action during repeat working,. The change of observation is modeled as the state space reduction using a formulation of the reinforcement learning. The state-reduction method enable to approach to a so-called frame problem by using recognition.

For the modeling, this study uses a dynamical system that is a finite Markov decision process (MDP) with discrete time. It then proposes a concept of state reduction, where the state space can be reduced conserving the optimal policy. An algorithm of the state space reduction is proposed. The method reduces the state space and accelerates the reinforcement learning.

As a concrete example, we consider a simple task modified from a Peg-in-Hole task of a manipulator. It is assumed that the learning robot does not know beforehand what state space is necessary and sufficient for the learning. If the dimensions of the state space are high, learning convergence needs very long time. The proposed method reduces the state space and the reinforcement learning may become applicable.

II. Reinforcement Learning

A. DP Formulation for Learning

1) General DP Problem: The reinforcement learning is associated with the dynamic programming (DP).

Generally, we have a discrete-time dynamic system in DP formulations [4]. If we are in state i and we choose action or control u, we will move to state j with probability $p_{ij}(u)$. When $p_{ij}(u)$ of transition to j is dependent on only current i and u, the system satisfies the Markov property and is called a Markov decision process (MDP). Time is occasionally considered as an element of the state. But, the systems considered here are not obviously dependent upon time.

The control u depends on the state i and the rule by which we select the controls is called a policy or feedback control policy. Simultaneously with a transition from i to j under control u, we incur a cost $g(i, u, j)$.

2) State, Control, and Policy: In a general reinforcement learning problem, state i, j and control u are discrete variables that are elements of the finite sets. There are n states, denoted by $1, 2, \ldots, n$, plus possibly an additional termination state, denoted by 0. At state i, the control must be chosen from a given finite set $U(i)$. At state i, the choice of a control u specifies the transition probability $p_{ij}(u)$ to the next state j.

We are interested in policies, i.e., sequence $\pi = \{\mu_0, \mu_1, \ldots\}$, where μ_k is to map each state i into a control $\mu_k(i) \in U(i)$. Let us denote by i_k the state at time k. Once a policy π is fixed, the sequence of states i_k becomes a Markov chain with transition probabilities:

$$P(i_{k+1} = j | i_k = i) = p_{ij}(\mu_k(i))$$

3) Problems and Costs: We can distinguish between finite horizon problems, where the cost accumulates over a finite number of stages N, and infinite horizon problems, where the cost accumulates indefinitely. The problems here are considered as infinite horizon problems to be solved by learning. In infinite horizon problems, a cost accumulates additively over time. At the kth transition, we incur a cost $\alpha^k g(i,u,j)$, where g is a given function, and α is a scalar with $0 < \alpha \leq 1$, called the discount factor. The total expected cost starting from an initial state i and using a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ is

$$J^\pi(i) = \lim_{N\to\infty} E\left[\sum_{k=0}^{N-1} \alpha^k g(i_k, \mu_k(i_k), i_{k+1}) \middle| i_0 = i\right]$$

where the expectation is taken with respect to the probability distribution of the Markov chain $\{i_0, i_1, \ldots i_N\}$. This distribution depends on the initial state i_0 and the policy π, as discussed earlier. The optimal cost-to-go starting from state i is denoted by $J^*(i)$; that is

$$J^*(i) = \min_\pi J^\pi(i)$$

We view the costs $J^*(i)$, $i = 1, \ldots, n$, as the components of a vector J^* that is referred as the optimal cost-to-go vector.

4) Stochastic Shortest Path Problem: Most optimal policies in infinite horizon problems are stationary policies, which are time-independent policies of the form $\pi = \{\mu, \mu, \ldots\}$. For brevity, we refer to $\{\mu, \mu, \ldots\}$ as the stationary policy μ. The corresponding cost-to-go is denoted by $J^\mu(i)$. The vector J^μ that has components $J^\mu(i)$, $i = 1, \ldots, n$, is referred to as the cost-to-go vector of the stationary policy μ. We say that μ is optimal if $J^\mu(i) = J^*(i)$ for all states i. The optimal infinite horizon cost-to-go satisfies the following form for all states i

$$J^*(i) = \min_\mu \sum_{j=1}^{n} p_{ij}(\mu)(g(i,\mu(i),j) + \alpha J^*(j)) \quad (1)$$

In this study, the discussed problems can be considered as stochastic shortest path problems that are a class of the infinite horizon problems. In the problems below, we assure that $\alpha = 1$ but there is an additional state 0, which is a cost-free termination state. Once the system reaches that state, it remains there at no further cost.

$$P_{00}(u) = 1, \quad g(0,u,0) = 0, \quad {}^\forall u \in U(0)$$

Under those conditions, we minimize the cost-to-go $J^\pi(i)$. We are interested in problems where reaching the termination state is in-evitable, at least under an optimal policy. Thus, the essence of the problem is how to reach the termination state with minimum expected cost.

B. Methods To Solve DP Problems

A value iteration and a Q-learning are introduced below to solve the above DP problems.

1) Value Iteration: For any vector $J = (J(1), \ldots, J(n))$, we consider the vector TJ obtained by applying one iteration of the DP algorithm to J; the components of TJ are

$$(TJ)(i) \quad = \quad \min_{u\in U} \sum_{j=0}^{n} p_{ij}(u)(g(i,u,j) + J(j)) \quad (2)$$

for $i = 1, \ldots, n$, where we will always use the convention that $J(0) = 0$. Note that T can be viewed as a mapping that transforms a vector J into the vector TJ. Furthermore, TJ is the optimal cost-to-go vector for a one-stage problem that has one-stage cost g and terminal cost J.

We will denote by T^k the composition of the mapping T with itself k times; that is, for all k we write

$$(T^k J)(i) \quad = \quad (T(T^{k-1}J))(i), \quad i = 1, \ldots, n$$

Thus, $T^k J$ is the vector obtained by applying the mapping T to the vector $T^{k-1}J$. For convenience, we also write $(T^0 J)(i) = J(i)$, $i = 1, \ldots, n$. It can be seen that $(T^k J)(i)$ is the optimal cost-to-go for the k-stage stochastic shortest path problem with initial state i, one-stage cost g, and terminal cost J.

Consider the stochastic shortest path problem under proper assumptions, and we have [4], [5]

$$\lim_{k\to\infty} T^k J \quad = \quad J^*$$

for every vector J. Based on the above equation, the DP iteration that generates the sequence $T^k J$ starting from some J is called value iteration and is a principal method for calculating the optimal cost-to-go vector J^*.

Generally, the method requires an infinite number of iterations. However, under special circumstances, the method can terminate finitely. A prominent example is the case of a deterministic shortest path problem. The initial condition $J(i) = \infty$ for $\forall i$ should be used when the learning algorithm starts, where a very large number is practically enough instead of ∞. This initial condition guarantees that each element of the cost-to-go vector monotonically decreases and converges to a true value. Moreover, in case of the value iteration, the cost-to-go vector converges to the optimal J^* within at most n times iteration [5].

2) Q-Learning: Above-mentioned J^* is calculated by Q-learning [2] that is a typical method of reinforcement learning. The Q-learning algorithm estimates the optimal action-value function $Q(i,u)$ through interactions between the robot and the environment with trial-and-error processes. The robot takes action u, observes new state j and cost $g(i,\, u,\, j)$, and updates Q as

$$Q(i,u) \leftarrow$$
$$(1-\gamma)Q(i,u) + \gamma[g(i,u,j) + \alpha \min_{u'} Q(j,u')] \quad (3)$$

where γ $(0 < \gamma \leq 1)$ and α $(0 < \alpha \leq 1)$ are a learning rate and a discount rate, respectively. The action-value function Q generates the cost-to-go $J(i)$ as

$$J(i) = \min_u Q(i,u)$$

It has been shown that the estimated Q converges to the optimal if the system is modeled as a finite Markov decision process and all actions are chosen enough times. To choose the action appropriately through learning, this study uses the ϵ-greedy policy [2] where any action is selected randomly with probability ϵ, otherwise the optimal action is chosen by using the current estimated $Q(i,\, u)$. This initial condition $J(i) = \infty$ for $\forall i$ guarantees that each

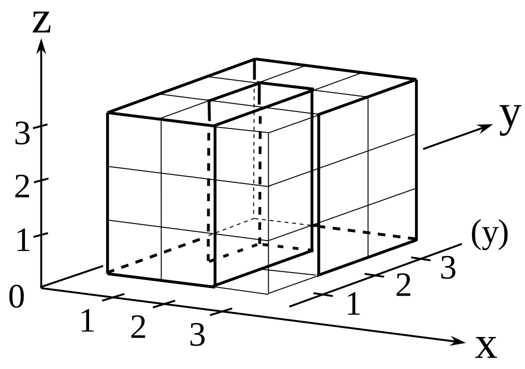

Fig. 1. A maze example for state reduction

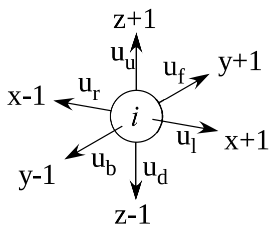

Fig. 2. Motions corresponding to controls

element of the cost-to-go vector monotonically decreases and converges to a true value as well as the value iteration case.

III. State Reduction

A. State Reduction Problem

The change of observation is found in the skill-acquiring human operators, whose environmental observation change efficiently as they repeat working. The change of observation can be modeled by the state reduction as follows, where minimum observation should be used to achieve the task.

If the same control input is applied under a policy in two or more states in state space, it is not necessary to distinguish those states. Therefore, they can be regarded as the same state. As a result, we consider that those states are same and can be reduced into one. We now will discuss the next problem.

Problem 1: Decide the lower dimensiona state space that holds the learning convergence and same optimal policy of the original state space.

B. Maze Problem

For explanation, consider the maze problem illustrated in Fig. 1 as a stochastic shortest path problem. Sensory information is available for x, y, z coordinates of the agent moving inside the maze and we consider $i = (i_x,\, i_y,\, i_z)^T$ as its state. Total number of the system states is $3^3 = 27$ because three positions of 1, 2, 3 are taken in each x, y, z component. There are three terminal states, i.e., goal states $i_G = (3,\, 1,\, i_z)^T$, where i_z is arbitrary. Therefore, there are $n = 27 - 3 = 24$ states with the exception of the terminal states. In Q-learning, the agent's initial state is chosen at random among the 24 states. Consider 6 kinds of control input u as $\{u_l, u_r, u_f, u_b, u_u, u_d\}$ and $g(i,\, u,\, j) = 1$ is incurred by each control. Fig. 2 shows the resultant motions by the control when no wall is adjacent to the agent. The agent cannot move when the moving direction is bordered by a wall, but is incurred by the control cost. The terminal states use $Q(i_G, u) = 0$ and $J(i_G) = 0$, and others start with $Q_0 = 100$ and $J_0 = 100$. The parameters used for Q-learning are a learning rate $\gamma = 0.6$ and a discount rate $\alpha = 1$ (with no discount).

C. Policy-Invariant State Reduction

In the above-mentioned problem, we evaluate optimal cost-to-go vector J^*_{xyz} and optimal policy μ^*_{xyz} using value iteration and Q-learning, respectively. Now, $J^*_{xyz}(i)$ and $\mu^*_{xyz}(i)$ at state $i = (i_x,\, i_y,\, i_z)^T$ are represented by $J^*_{xyz}(i_x,\, i_y,\, i_z)$ and $\mu^*_{xyz}(i_x,\, i_y,\, i_z)$, respectively. In this example, both $J^*_{xyz}(i)$ and $\mu^*_{xyz}(i)$ are independent from i_z. Therefore, the optimal cost-to-go vector J^*_{xyz} and the optimal policy μ^*_{xyz}, respectively, are given by matrices whose elements at i_x-th row and i_y-th column are $J^*_{xyz}(i_x,\, i_y,\, i_z)$ and $\mu^*_{xyz}(i_x,\, i_y,\, i_z)$ as:

$$J^*_{xyz} = \begin{bmatrix} 6 & 5 & 4 \\ 7 & 8 & 3 \\ 0 & 1 & 2 \end{bmatrix},\quad \mu^*_{xyz} = \begin{bmatrix} u_f & u_f & u_r \\ u_l & u_b & u_r \\ 0 & u_b & u_b \end{bmatrix} \quad \text{for } \forall i_z \tag{4}$$

The element $J^*_{xyz}(i)$ indicates how many steps the optimal policy will need from state i to the terminal state because the control cost for a step is 1.

In this problem, because $J^*_{xyz}(i)$ is independent from i_z, there is J^*_{xy} dependent on only i_x and i_y satisfying

$$\begin{aligned} J^*_{xyz}(i_x,\, i_y,\, i_z) &= J^*_{xy}(i_x,\, i_y) \ \text{ for } \forall i_z \\ \mu^*_{xyz}(i_x,\, i_y,\, i_z) &= \mu^*_{xy}(i_x,\, i_y) \ \text{ for } \forall i_z \end{aligned} \tag{5}$$

Consequently, the actions induced by the state $(i_x, i_y, i_z)^T$ and the optimal policy μ^*_{xyz} are the same as those by $(i_x,\, i_y)^T$ and μ^*_{xy}.

In this situation, we say for the maze problem that "the optimal policy μ^*_{xyz} for state $(i_x,\, i_y,\, i_z)^T$ is equivalent to the optimal policy μ^*_{xy} for state $(i_x,\, i_y)^T$." We also say that "the state space of $(i_x, i_y, i_z)^T$ can be reduced to that of $(i_x,\, i_y)^T$ conserving the optimal policy μ^*_{xyz}."

D. State Reduction by Coordinate Exclusion

Consider a vector space with coordinates for the state space, e.g., the state $(i_x,\, i_y,\, i_z)^T$ in the maze example. In addition, we make a reduced state space by excluding coordinates from the original state space in the same manner that the low dimensional state space of $(i_x,\, i_y)^T$ is made by excluding z-coordinate from the original state.

As shown in this example, the optimal policy μ^*_{xy} and the optimal cost-to-go J^*_{xy} for the reduced state space of $(i_x,\, i_y)^T$ are obtained by the following equations when the original state space can be reduced by excluding i_z from the original state when J^*_{xyz} is independent from i_z.

$$\begin{aligned}
&\underline{i_z} = \arg\min_{i_z} J^*_{xyz}(i_x,\, i_y,\, i_z),\\
&J^*_{xy}(i_x,\, i_y) = J^*_{xyz}(i_x,\, i_y,\, \underline{i_z}),\\
&\mu^*_{xy}(i_x,\, i_y) = \mu^*_{xyz}(i_x,\, i_y,\, \underline{i_z}) \;\text{ for } \forall i_x,\, \forall i_y
\end{aligned} \tag{6}$$

The optimal cost-to-go J^*_{xyz} is constant along with z-axis, but the minimum operation with respect to i_z is for a later application.

In the maze example, a control of a deterministic policy is dependent on i_x and i_y. Hence, we cannot acquire a proper policy using a low dimensional space without i_x or i_y. No deterministic policy is proper for low dimensional state spaces $(i_y,\, i_z)^T$, $(i_x,\, i_z)^T$, (i_x), (i_y), and (i_z). For those low dimensional state spaces, we can formally perform calculations similar to Eq. (6), but the results are not the optimal policy and optimal cost-to-go.

The policy and cost-to-go obtained formally by Eq. (6) is optimal if the cost-to-go calculated by the policy evaluation step does not change. Practically, the value iteration with the obtained policy may be used for the policy evaluation step.

Remark 1: Suppose that the optimal policy μ^* and the optimal cost-to-go J^* are independent from a coordinate in the state space with coordinates. A low order state space is constructed by excluding the coordinate, and the optimal policy and the optimal cost-to-go can be calculated as well as Eq. (6). Therefore, the original state space can be reduced conserving the optimal policy.

E. Algorithm to Find Low Dimensional State

The following algorithm is considered in order to find a reduced state space before the learning converges and the optimal cost-to-go is determined in the original state space.

1) The cost-to-go J_{xy} and the policy μ_{xy} for the candidate of low dimensional state space are calculated by Eqs. (6) using the cost-to-go J_{xyz} and the policy μ_{xyz} obtained at the moment. The $J_{yz}, J_{zx}, \ldots, J_z$, etc. are calculated similarly.
2) When a pare among $(J^*_x,\, \mu^*_x), \ldots, (J^*_{xyz},\, \mu^*_{xyz})$ is converge, we evaluate whether the acquired policies are the optimal. Generically, the acquired policy is the optimal when it is feasible, and the original state space can be reduced conserving the optimal policy.

F. Numerical Result

Because of the minimum operation in Eqs. (6), J_{xy} converges earlier than J_{xyz}, or simultaneously at the latest. In this problem, the state number of $(i_x,\, i_y,\, i_z)$ is 3 times as many as those of $(i_x,\, i_y)$ since i_z has 3 states. Therefore, J^*_{xyz} takes 3 times as long as J^*_{xy} takes for convergence when it uses the value iteration. On the other hand, the result of Q-learning is shown in Fig. 3. The vertical axis and the horizontal axis respectively show numbers of episodes for Q_{xyz} and Q_{xy}, whose learning convergence are judged by the same convergence criteria. In case of Q-learning, Q_{xyz} takes between 30 and 400 times as long as Q_{xy} takes for convergence whereas the factor of their state numbers is only 3. The average factor of 200 simulations is about 130. Hence, the proposed method is especially effective in the Q-learning with trial-and-error processes.

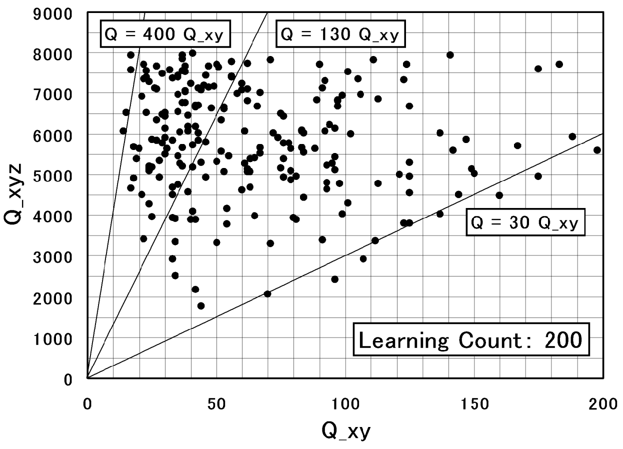

Fig. 3. Episode numbers of Q-learning by using Q_{xy} and Q_{xyz}

IV. Manipulator Problem

A. Problem Definition

As shown in Fig. 4, we consider a task where a 3-link SCARA type manipulator places a circular component into a hole. This is simplified from the Peg-in-Hole task and no friction is contained for simplicity.

It is reasonable to use the state variables of the equations of motion of the robot manipulator with the geometric endpoint constraint as:

$$i = (x, y, \alpha, \dot{x}, \dot{y}, \dot{\alpha}, f_x, f_y, n_z)^T \tag{7}$$

where the elements are x, y positions, $\alpha = \theta_1 + \theta_2 + \theta_3$ direction of the hand with component, their velocities, and applied forces in x, y, α directions from the environment, respectively. They all are measured from the sensors. The initial state is $x = 0.28, y = 0.28, f_x = 0$, and $f_y = 0$. The terminal state is $x_d = 0.40, y_d = 0.40, f_{xd} = 0$, and $f_{yd} = 0$. The x and y coordinates are descretized every 0.03[m], α every 10[deg], and f_x and f_y every 0.7[N]. A state is then described by a set of integers. For example, it expresses $x = 0.03(i_x - 1) + 0.28$, $i_x = 1, 2, \ldots, 9$ and i_x is considered as x. Other states are also descretized. The $\dot{x}, \dot{y}, \dot{\alpha}$, and n_z are always zeros in this particular example.

The following controller is used to generate actions:

$$\boldsymbol{\tau} = -\boldsymbol{J}^T \boldsymbol{K}_P(\boldsymbol{y} - \boldsymbol{y}_r) - \boldsymbol{K}_D\dot{\boldsymbol{\theta}} \tag{8}$$

where $\boldsymbol{\tau}$ is control input to the manipulator, $\boldsymbol{J} = \partial \boldsymbol{y}/\partial \boldsymbol{q}^T$ Jacobian matrix, $\boldsymbol{y} = [x, y, \alpha]^T$ manipulation variable vector, $\boldsymbol{y}_r$ reference of $\boldsymbol{y}$, $\boldsymbol{\theta}$ joint variable vector, $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ feedback gain matrices, respectively. For the reference manipulation variable $\boldsymbol{y}_r^{(k)}$ at time k, $\boldsymbol{y}_r^{(k+1)}$ is given by

$$\boldsymbol{y}_r^{(k+1)} = \boldsymbol{y}_r^{(k)} + \delta \boldsymbol{y}_r^{(k)} \tag{9}$$

Control at time k is considered as the $\delta \boldsymbol{y}_r^{(k)}$. The $\delta \boldsymbol{y}_r^{(k)}$ is constant during the period from time k to time $k+1$. Fig. 5

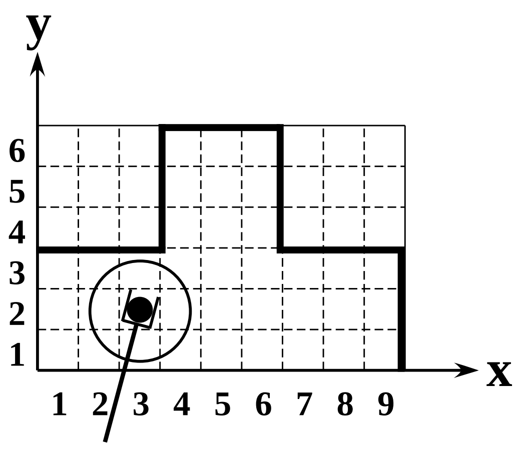

Fig. 4. Simplified Peg-in-Hole

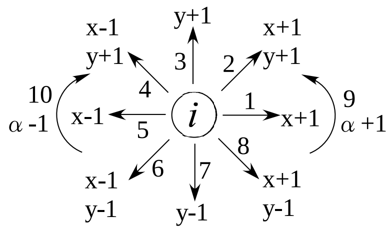

Fig. 5. Hand motions correspond to controls

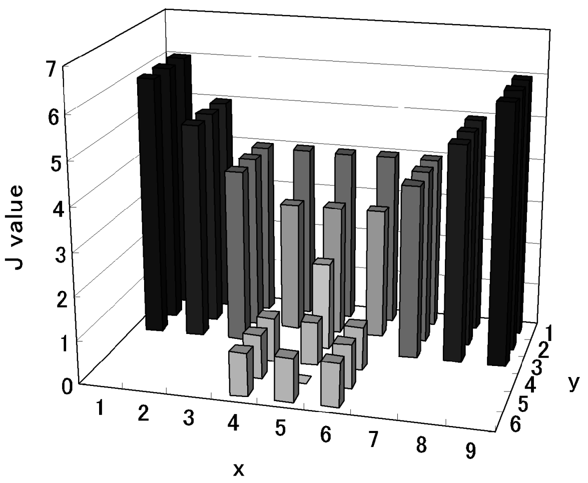

Fig. 6. Cost distribution of $J^*_{4-1}(x, y, f_x, f_y)$

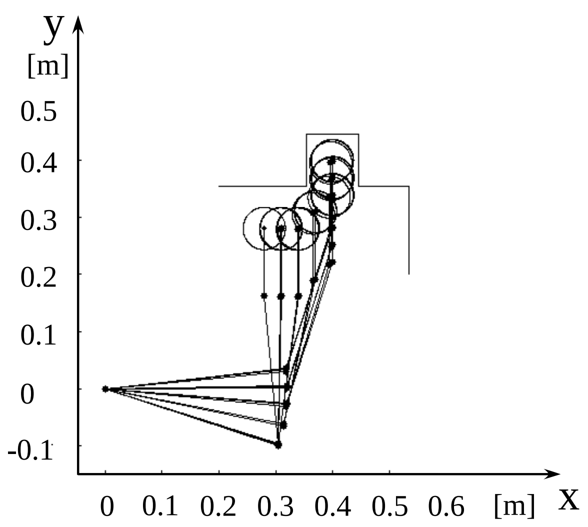

Fig. 7. Acquired motion started from $(0.28,\ 0.28)$

shows the ten kinds of control that the manipulator can use. For each control, cost $g(i,\ u,\ j) = 1$ is incurred evenif the hand with component cannot move against the walls. When a reference value $\boldsymbol{y}_r$ goes out of state space, it remains at the previous position and a large cost is incurred.

For example, state and the reference are $(x,\ y,\ \alpha,\ f_x, f_y)^T = (8, 2, 1, 2, 2)^T$ and $\boldsymbol{y}_r^{(k)} = (8, 2, 1)^T$ in time k. The state will become $(x, y, \alpha, f_x, f_y)^T = (8, 2, 1, 3, 3)^T$ at time $k+1$ if $\boldsymbol{y}_r^{(k+1)} = (\ 9,\ 3,\ 1\ )^T$.

The terminal state is $i_G = (5,\ 5,\ \alpha,\ 1,\ 1,\ 1,\ 2,\ 2,\ 1)$ and α is arbitrary. The terminal stats use $Q(i_G, u) = 0$, $J(i_G) = 0$ and others start with $Q_0 = 20$, $J_0 = 20$. The parameters used for Q-learning are taken as a learning rate $\gamma = 0.9$ and a discount rate $\alpha = 1$ (with no discount).

B. State Reduction

The optimal cost-to-go vector and the optimal policy are expressed as J^*_9 and μ^*_9 for the original 9-dimensional state space. The J^*_9 and μ^*_9 are evaluated by using Q-learning. The original state space is reduced conserving the optimal policy by using the same algorithm as the previous section.

There are J^*_{4-1} and μ^*_{4-1} dependent on only x, y, f_x, and f_y because $J^*_9(i)$ is independent from α, and $\dot{x}, \dot{y}, \dot{\alpha}$, and n_z are constant in this particular example. Therefore, the following equations hold:

$$\begin{aligned} J^*_{4-1}(x, y, f_x, f_y) &= J^*_9(x, y, \alpha, \dot{x}, \dot{y}, \dot{\alpha}, f_x, f_y, n_z) \\ &\qquad \text{for } \forall\alpha \\ \mu^*_{4-1}(x, y, f_x, f_y) &= \mu^*_9(x, y, \alpha, \dot{x}, \dot{y}, \dot{\alpha}, f_x, f_y, n_z) \\ &\qquad \text{for } \forall\alpha \end{aligned} \tag{10}$$

We say for this problem that the optimal policy μ^*_9 is equivalent to the optimal policy μ^*_{4-1}. We also say that the 9-dimensional state space can be reduced to 4-dimensional state space conserving the optimal policy μ^*_9.

C. Simulation Result

For Q-learning, the same algorithm as the maze problem achieves state reduction. The distribution of the cost-to-go $J^*_{4-1}(x, y, f_x, f_y)$ is shown in Fig. 6. The J^*_9 takes about 30 or more times as long as J^*_{4-1} takes for convergence. The reduced state space enable to converge Q-learning and obtained optimal manipulator actions are shown in Figs. 7 and 8. Fig. 7 is the optimal action from state $(x,\ y,\ f_x,\ f_y) = (1,\ 1,\ 2,\ 2)$ to terminal state $(x, y, f_x, f_y) = (5, 5, 2, 2)$, where the manipulator reaches terminal state in 6 steps. Fig. 8 is the optimal action from state $(x,\ y,\ f_x,\ f_y) = (8,\ 1,\ 2,\ 2)$, and it reaches terminal state in 5 steps.

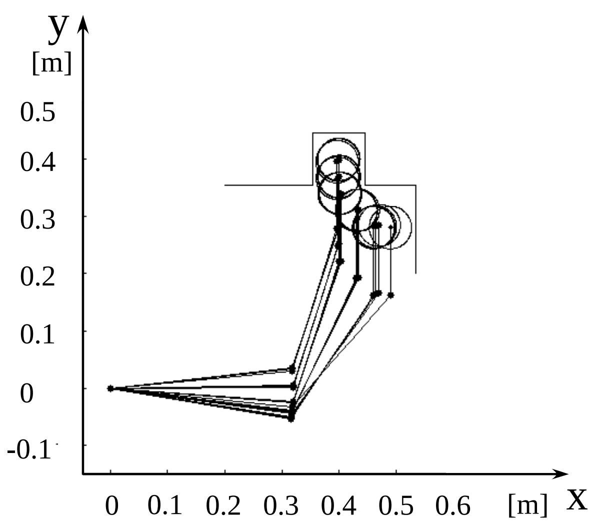

Fig. 8. Acquired motion started from $(0.49,\ 0.28)$

V. Concluding Remarks

The concept has been defined that the state space can be reduced conserving the optimal policy. For the case that the reduced state space can be constructed by coordinates exclusion, the algorithm has been given that the optimal cost-to-go and the optimal policy of the reduced space is calculated from those of the original space. Its usefulness for DP iteration and Q-learning has been compared. For DP iteration, the convergence of the optimal cost-to-go in the original state space has needed approximately N times as long as that in the reduced state space, where N is a ratio of the number of the original states to the reduced. On the other hand, numerical examples for Q-learning has shown that the one in the original state space needs for more than N times as long as that in the reduced state space. Hence, the proposed state space reduction method can accelerate those learning methods. The acceleration effect for Q-learning has been more remarkable than that for the DP iteration. The state space reduction can be considered as a model of the change of observation. The obtained results have explained that the change of observation is reasonable in terms of learning efficiency. The significance of the proposed concepts has been clarified simultaneously.

There remain some subjects for autonomous space robots. The approach to the autonomy and/or intelligence is the biggest subject to realize useful space robots. This study has approached this issue by the reinforcement learning algorithm, whereas there remain many research subjects. Refer to [1] for the details of the subjects.

A part of this work was financially supported by a Grant-in-Aid for Scientific Research from Research Foundation for the Electrotechnology of Chubu and Ministry of Education, Science, Culture and Sports of Japan.

References

[1] Senda, K., Matumoto, T., Okano, Y., and Mano, S., "A Study toward An Autonomous Space Robot", *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, May 19–23, 2003, Nara, Japan, AS-9, pp. 1–8.

[2] Sutton, R. S. and Barto, A., *Reinforcement Learning*, MIT Press, Cambridge, MA, 1998.

[3] Sawaragi, T., "Proficient Skills Embedded with in Human, Machine and Environment," *J. Society of Instrument and Control Engineers*, vol. 37, no. 7, pp. 471-476, 1998. (in Japanese)

[4] Bertsekas, D. P. and Tsitsiklis, J. N., *Neuro-Dynamic Programming*, Athena Scientific, Belmont, 1996.

[5] Bertsekas, D. P., *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1995.