Stochastic Real-Valued Reinforcement Learning to Solve a non-Linear Control Problem

H. Kimura & S. Kobayashi*

***Tokyo Institute of Technology**
Department of Computational Intelligence and Systems Science,
Interdisciplinary Graduate School of Science and Engineering,
4259, Nagatsuta, Midori-ku, Yokohama, 226–8502 JAPAN
gen@fe.dis.titech.ac.jp, kobayasi@dis.titech.ac.jp

ABSTRACT

This paper presents a new approach to reinforcement learning (RL) to solve a non-linear control problem efficiently in which state and action spaces are continuous. Many DP-based reinforcement learning (RL) algorithms approximate the value function and give a greedy policy with respect to the learned value function. However, it is too expensive to fit highly accurate value functions, particularly in continuous state-action spaces. We provide a hierarchical RL algorithm composed of local linear controllers and TD-learning, which are both very simple. The continuous state space is discretized into an array of coarse boxes, and each box has its own local linear controller for choosing primitive continuous actions. The higher-level of the hierarchy accumulates state-values using tables with one entry for each box. Each linear controller improves the local control policy by using an actor-critic method. The algorithm was applied to a simulation of a cart-pole swing-up problem, and feasible solutions are found in less time than those of conventional discrete RL methods.

1 INTRODUCTION

Many DP-based reinforcement learning (RL) algorithms approximate the value function and give a greedy policy with respect to the learned value function. Theoretical results guarantee that several DP-based algorithms will find optimal policies (e.g. [16], [15], etc.) and a great deal of effort has been made on the techniques to approximate value functions (e.g., CMAC, Neural-Net). However, it is too expensive to fit highly accurate value functions, particularly in continuous state-action spaces. To overcome this problem, several techniques are proposed (e.g., Parti-game [10], Coarse grids +environment models +search techniques [4], interpolation on a coarse grid [3]). On the other hand, it is shown that a local gradient-ascent search over stochastic policy-space is possible without explicitly computing value or gradient estimates [17], [8]. That is to say, the DP-based RL methods have features of a global search in terms of optimality and computational expense, and the gradient RL methods have a feature of a local search. This paper presents a new model-free approach that bridges a gap between DP-based methods and local gradient-ascent methods.

In real-world applications, an approach combining discrete RL methods with linear controllers is promising since there are many non-linear control problems that can be decomposed into several local linear control tasks. Based on this principle, we provide a new algorithm with the key ideas including:

- A hierarchical RL method composed of discrete TD method [13] and local linear controllers, which are both very simple.
- Generalization techniques for continuous state-action spaces: *coarse discretization* for the high-level of the hierarchy, and *local continuous-vector* representation for the low-level linear controllers.
- A policy improvement algorithm for the local linear controllers with imperfect value functions.

The coarse state-space quantization is a quite simple way to cope with the curse of dimensionality. But in many cases, it often gives rise to undesirable non-Markovian effects. In our approach, local linear controllers, which use a continuous-vector for the local state representation, make up for these effects.

Although hierarchical algorithms can often yield sub-optimal solutions, there are significant benefits in learning efficiency, search space, and re-use of knowledge [5], [14], [12]. In general, the low-level of the hierarchy is composed of learning control modules that represent subtasks or abstract actions. Each subtask is defined in terms of termination conditions, and the low-level module is to find a local optimal policy. The RL techniques for discrete semi-Markov decision processes (SMDPs) would be applied to learning of the high-level module that selects a low-level modules as abstract action. In our approach, the low-level modules correspond to linear controllers.

Actor-critic methods [1] are often used for learning on the linear controllers [7], [6]. Generic actor-critic methods can be classified into a local gradient-ascent method with respect to policy space by using learned value functions. But in our hierarchy, the linear controllers cannot hold accurate value functions on account of practical limitations on the computational resources. Fortunately, it is shown that an actor-critic

0-7803-5731-0/99/$10.00 ©1999 IEEE

algorithm using eligibility traces in the actor can improve its stochastic policy even though the estimated value function is inaccurate [8]. We take advantage of this method for the local policy improvement.

2 PROBLEM FORMULATION

2.1 Markov Decision Processes

At each discrete time t, the agent observes x_t containing information about its current state, select action a_t, and then receives an instantaneous reward r_t resulting from state transition in the environment. In general, the reward and the next state may be random, but their probability distributions are assumed to depend only on x_t and a_t in Markov decision processes (MDPs), in which many reinforcement learning algorithms are studied. In MDPs, the next state y is chosen according to the transition probability p^a_{xy}, and the reward is given randomly according to the expectation r^a_x. but the agent does not know p^a_{xy} and r^a_x ahead of time. The objective of reinforcement learning is to construct a policy that maximizes the agent's performance. A natural performance measure for infinite horizon tasks is the cumulative discounted reward:

$$V_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \ , \qquad (1)$$

where the discount factor, $0 \leq \gamma < 1$ specifies the importance of future rewards, and V_t is the value on time t. In MDPs, the value can be defined as:

$$V^{\pi}(x) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k \, r_k | x_0 = x\right] \ , \qquad (2)$$

where E_{π} denotes the expectation assuming the agent always uses stationary policy π. The objective in MDPs is to find an optimal policy that maximizes the value of each state x defined by Equation 2. In MDPs, the optimal value in state x, denoted $V^*(x)$ satisfies the Bellman equations for all x:

$$V^*(x) = \max_a \left[r^a_x + \sum_y p^a(x,y) + \gamma \, V^*(y) \right] \ . \qquad (3)$$

2.2 A Non-Linear Control Problem

We are given learning control of non-linear dynamic systems in which :

- State and action spaces are continuous and multi-dimensional.
- The task could be decomposed into several local linear or bang-bang control tasks.
- The agent does not know dynamics of the environment ahead of time.

Throughout the paper, we use a cart-pole swing-up task for the example.

3 COMBINING DISCRETE TD(0) WITH LOCAL LINEAR CONTROLLERS

3.1 Hierarchical Decomposition

In our approach, the agent adopts hierarchical state representation. The higher-level of the hierarchy uses discrete representation of the state variables by coarse quantization. Representative points are situated at the center of the grid's boxes. Each box has its own local linear controller.

Assume that given a n-dimensional state input $(x_1, x_2, \cdots x_n)$ and the corresponding box (B_i) with the representative point at $(b^i_1, b^i_2, \cdots b^i_n)$, then the low-level linear controller gets a local continuous input

$$C = (c_1, c_2 \cdots c_n) = (x_1 - b^i_1, x_2 - b^i_2, \cdots, x_n - b^i_n) \ . \qquad (4)$$

Figure 1 shows an example of the state representation in which the state space is two-dimensional.

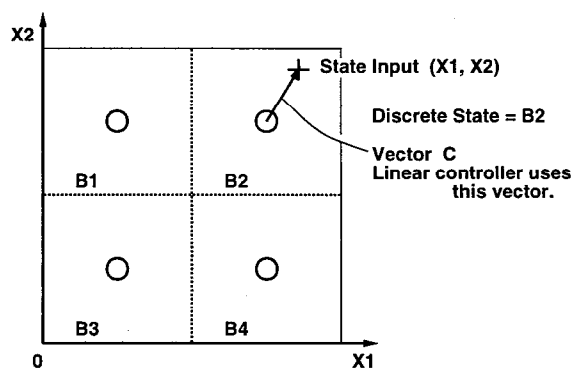

Figure 1: An example of the hierarchical state representation in a 2-dimensional task. Small circles at the center of the grid's boxes denote its representative points.

3.2 Event-driven Decision Making

In dynamic control, an optimal policy often selects the same action (or holds similar continuous action) for a certain period of time. Then, uniform regions are likely to exist in the state space where all of the states have the same (or similar) action. In our approach, the coarse grids approximate the uniform regions. The high-level decision maker selects abstract action each time when the continuous state input is going across the boundary of the box, or a certain period of time passes. This model of abstract action can be seen as an extension of Markov options [14] in which the policy is given by linear controller and the termination condition is given by the boundary of the box. Figure 2 helps to define the big-steps that the high-level learner takes. If all the linear controllers hold stationary policy, the high-level learner is to solve a semi-Markov decision problem.

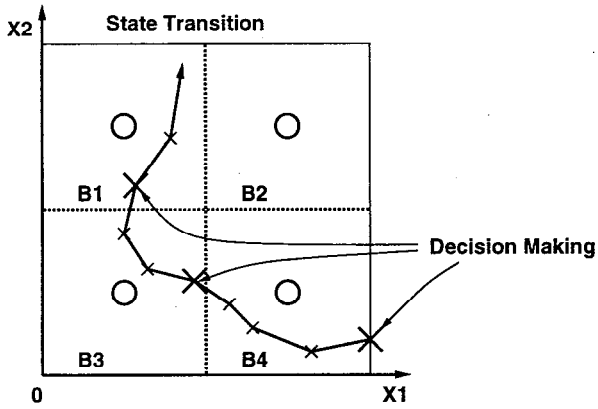

Figure 2: The big-steps on the high-level learner. This modeling is closely related to semi-Markov decision processes (SMDPs). The high-level decisions are only allowed at events which occur whenever the state transition goes across the boundary of the box, or a certain period of time passes.

3.3 A Learning Algorithm

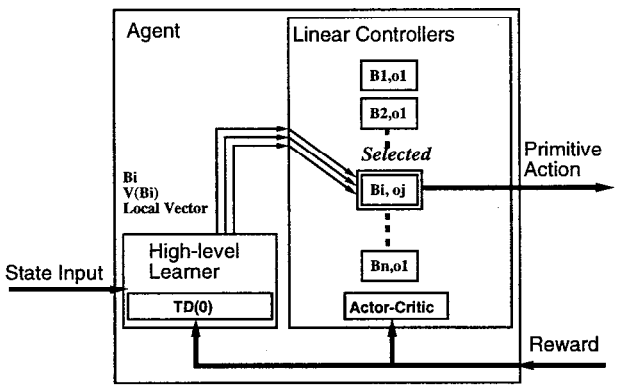

Figure 3: The hierarchical structure of the algorithm. $B_1, B_2, \cdots B_i \cdots B_n$ denote the state boxes, and every small rectangle labeled "B_*, o_1" denotes a linear controller.

Figure 3 gives an overview of the hierarchical algorithm. The high-level learner accumulates state values using tables with one entry for each box. In this case, each box value $V(B_i)$ represents approximation of averaged values weighted by the visiting frequency over the coarse regions, i.e., $V(B_i) \simeq \sum_{x \in B_i} U^\pi(x) V^\pi(x)$, where B_i denotes the box, $U^\pi(x)$ denotes the probability of occupying state x under the policy π. The learner also accumulates state value $V(B_i)$. Unfortunately, this value function does not satisfy Bellman equations in the strict sense. However, we try to provide a good approximation algorithm.

Assume that the agent selects a linear controller at time t in the region $B(t)$ and makes next decision at time $t+k+1$ in the region $B(t+k+1)$. Then we approximate values at each time by:

$$\begin{aligned} E\{V_t^\pi\} &\simeq E\{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots + \gamma^k r_{t+k}\} \\ &\quad + \gamma^{k+1} V(B(t+k+1)), \\ E\{V_{t+1}^\pi\} &\simeq E\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots + \gamma^{k-1} r_{t+k}\} \\ &\quad + \gamma^k V(B(t+k+1)), \\ &\vdots \\ E\{V_{t+k}^\pi\} &\simeq E\{r_{t+k}\} + \gamma V(B(t+k+1)). \end{aligned}$$

We approximate the value of the box B_i according to averaging these values:

$$V(B_i) = \frac{1}{k+1}\left(V_t^\pi + V_{t+1}^\pi \cdots + V_{t+k}^\pi\right). \qquad (5)$$

Then, the TD-error for $V(B)$ in the high-level module is given by

$$\text{TD-error} = \frac{1}{k+1}\left(\sum_{i=0}^{k} r_{t+i} \sum_{j=0}^{i} \gamma^j\right) + \frac{1}{k+1}\left(\sum_{i=0}^{k} \gamma^{i+1}\right) V(B(t+k+1)) - V(B(t)). \qquad (6)$$

This result leads us to the learning algorithm for the high-level learner.

A Learning Method for High-Level Module: To estimate box values, the algorithm updates by

$$V(B(t)) \leftarrow V(B(t)) + \alpha_v \,(\text{TD-error}), \qquad (7)$$

where α_v is a learning coefficient, and TD-error is given by (6). Although this algorithm is ad hoc, it works very well for a reason that the learning in the linear controllers makes up for undesirable effects.

A Learning Method for Linear Controllers: The high-level module selects a linear controller associated with the corresponding state box. When the state input vector is n-dimensional, the linear controllers have $n+1$ parameters $W = (w_1, w_2, \cdots w_{n+1})$ per dimension of the output vector. The controllers learn W to make good control. A linear controller determines its state feedback-gain parameters $(g_1, g_2, \cdots g_{n+1})$ as to the control rule π by using W, . After that, primitive actions are selected according to π until the termination. The linear controllers adopt an actor-critic architecture to learn feedback-gain parameters. The actor-critic provides continuous action as the feedback-gain, and makes use of $V(B_i)$ for the critic. The output of the critic is too coarse to calculate value gradient from the learned value function, because the approximation function $V(B_i)$ is flat in the same state region. Therefore we adopt a slightly modified actor-critic algorithm using eligibility traces in the actor [8], that can improve its policy even though the estimated value function is inaccurate.

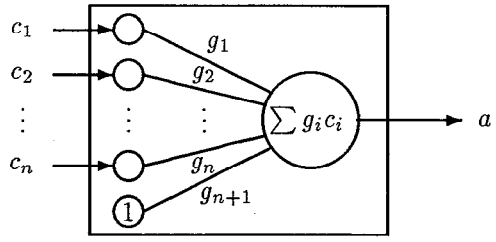

Figure 4: An instanceof the linear controller. g_i denotes a feedback gain which is characterized by w_i according to $g_i = w_i + N(0, \sigma^2)$. The controller learns w_i to achieve good control.

Figure 4 shows an instance of the actor-critic algorithm in which the action is of one dimension. In the beginning of the linear control, the actor selects its feedback-gain parameters $(g_1, g_2, \cdots g_{n+1})$ according to $g_i = w_i + N(0, \sigma^2)$ for all $i = 1, 2, \cdots n+1$, where $N(0, \sigma^2)$ denotes the normal distribution. At each time step $t < t+i \leq t+k$, primitive actions are selected until the termination of the linear control according to

$$a = c_1 g_1 + c_2 g_2 + \cdots + c_n g_n + g_{n+1} \ , \qquad (8)$$

where $C = (c_1, c_2, \cdots c_n)$ is the local continuous vector at the time $t+i$ as shown in Equation 4. When the linear controller terminates at time $t+k+1$ in the region $B(t+k+1)$, all linear controllers updates according to

$$\begin{aligned} Trace_i &\leftarrow Trace_i + (g_i - w_i) \\ w_i &\leftarrow w_i + \alpha_{ac}\,(\text{TD-error})\, Trace_i \\ Trace_i &\leftarrow \frac{1}{k+1}\Big(\sum_{j=0}^{k} \gamma^{j+1}\Big) Trace_i \end{aligned}$$

where $Trace_i$ denotes an eligibility trace on w_i, and α_{ac} is a coefficient of learning. TD-error is shown in Equation 6. The eligibility of w_i corresponds to $(g_i - w_i)$ that is similar to *Gaussian unit* [17] and Gullapalli's *neural reinforcement learning unit* [7].

4 Applying to a Cart-Pole Swing-Up Task

The behavior of this algorithm is demonstrated through a computer simulation of a cart-pole swing-up task. We modified the cart-pole problem described in [1] so that the action is taken to be continuous. The dynamics are modeled by

$$\ddot{\theta} = \frac{g \sin\theta + \cos\theta \left(\frac{-F - m\ell\dot{\theta}^2 \sin\theta + \mu_c sgn(\dot{x})}{M+m} \right) - \frac{\mu_p \dot{\theta}}{m\ell}}{\ell \left(\frac{4}{3} - \frac{m\cos^2\theta}{M+m} \right)} ,$$

$$\ddot{x} = \frac{F + m\ell \left(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta \right) - \mu_c sgn(\dot{x})}{M+m} ,$$

where $M = 1.0$(kg) denotes mass of the cart, $m = 0.1$(kg) is mass of the pole, $2\ell = 1$(m) is the length of the pole, $g = 9.8(m/sec^2)$ is the acceleration of gravity, F(N) denotes the force applied to cart's center of mass, $\mu_c = 0.05$ and $\mu_p = 0.01$ are coefficient of friction of the pole and the cart respectively. In this simulation, we use discrete-time system to approximate these equations, where $\Delta t = 0.02$sec. At each discrete time step, the agent observes $(x, \dot{x}, \theta, \dot{\theta})$, and controls the force F. The agent can execute action in arbitrary range, but the possible action in the cart-pole system is constrained to lie in the range $[-10, 10](N)$. When the agent chooses an action which does not lie in that range, the action F is saturated. The system begins with $(x, \dot{x}, \theta, \dot{\theta}) = (0, 0, 3.0, 0)$. When the cart collides with the end of the track $(-3.0 \leq x \leq 3.0)$, the cart rebounds from the bumper with a coefficient of rebound 0.2. The agent receives a reward (penalty) signal of
(a) -1 when the pole falls over $\pm 0.8\pi$ (rad),
(b) -4 when the cart bounces at the end of the track,
(c) -3 when $\dot{\theta} < -10$ or $10 < \dot{\theta}$ (rad/sec),
(d) $+1$ when $-0.133\pi < \theta < 0.133\pi$ and $-2 < \dot{\theta} < 2$ (rad/sec).

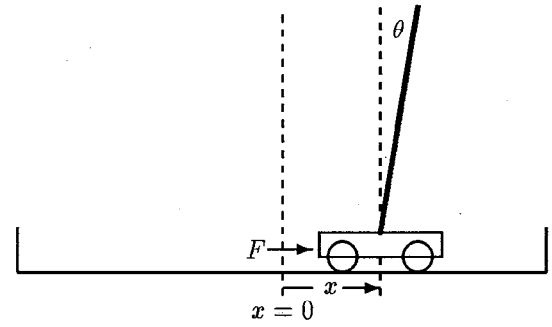

Figure 5: A simulation model of the cart-pole task.

4.1 Implementation

The state space is normalized as $(x, \dot{x}, \theta, \dot{\theta}) = (\pm 3.0$ m$, \pm 10$ m/sec$, \pm\pi$ rad$, \pm 10$ rad/sec$)$ into $(\pm 0.5, \pm 0.5, \pm 0.5, \pm 0.5)$. The agent discretizes the normalized state space evenly into $3 \times 3 \times 5 \times 5 = 225$ or $3 \times 3 \times 6 \times 6 = 324$ boxes, and attempts to store $V(B_i)$ in each box. Each linear controller accumulates five feedback-gain parameters $w_1, w_2, \cdots w_5$, for the continuous-state input is 4-dimensional. In the linear controllers, an action a is generated by Equation 8, where the normal distribution follows $\sigma = 0.5$. Then the force F is executed according to $F = a \times 20$.

The high-level hierarchy makes decision each time when the continuous state input moves to a different box or the state input keeps within the same box for 2 seconds (100 steps). The learning rate α_v is set to 0.3 in 3x3x6x6 boxes or to 0.1 in 3x3x5x5 boxes. The learning coefficient of the actor-critic is set to $\alpha_{ac} = 0.01$. The discount rate is set to $\gamma = 0.98$. Initial feedback-gain parameters are all set to zero for all linear controllers.

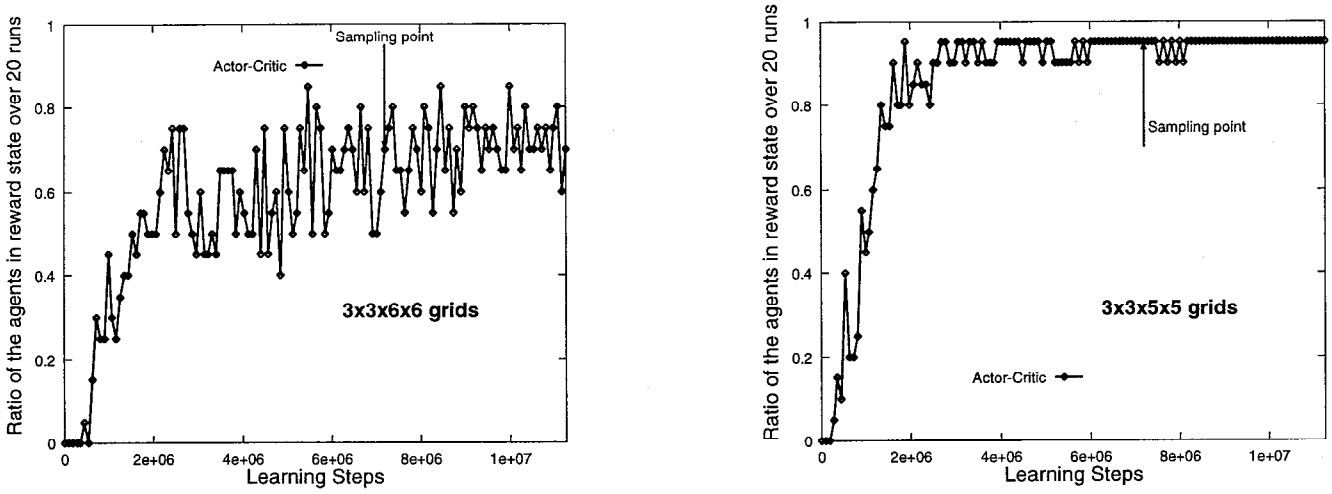

Figure 6: On-line performance using 3x3x6x6 and 3x3x5x5 grids.

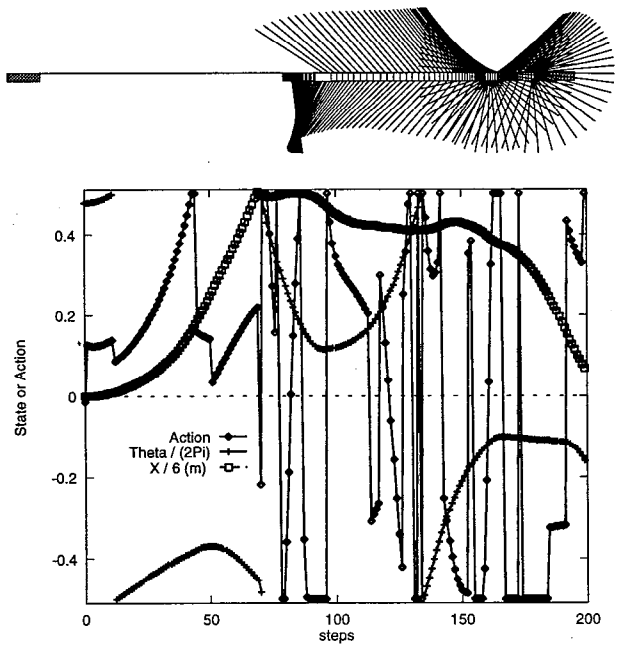

Figure 7: An example of behavior on 3x3x6x6 grids.

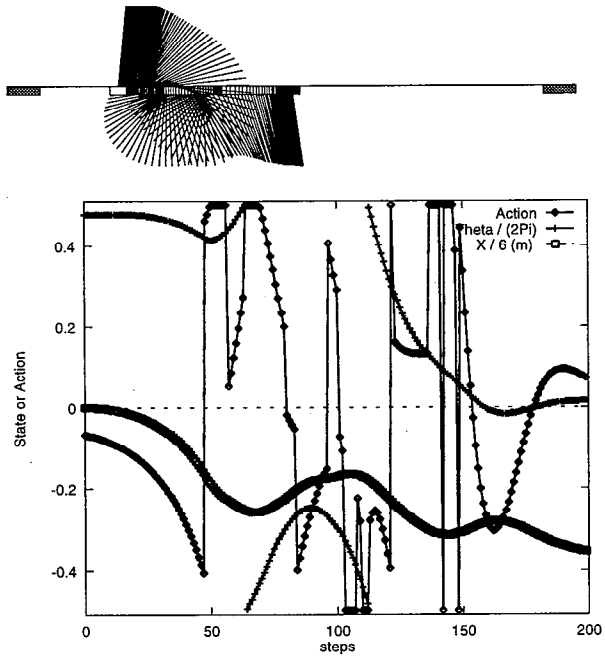

Figure 8: An example of behavior on 3x3x5x5 grids.

4.2 Simulation Results

Figure 6 shows the on-line performance with using different grids. The performance measure is the occupancy rate of the current state in which the system gives positive reward. The rate is calculated by using 20 independent runs. The results show that the proposed methods achieved learning to gain positive rewards. The increase of the performance with $3\times3\times5\times5$ boxes is better than with $3\times3\times6\times6$ boxes. One reason for this is an effect of decreasing the number of boxes by the coarse discretization. The other is that the location of the boundaries of the boxes fits for this task fortuitously. Figure 7 and 8 show examples of trajectories on learned greedy policies after 7,200,000 steps (40 hours on simulation time). The policies were feasible, but no agent could obtain optimal swing-up behavior. The reason is that the growth of swing-up behavior is slowed down for decreasing in opportunity of swing-up, because the agent makes progress in keeping the pole vertical. The bottom in figure 7 and 8 shows the corresponding state-action flow respectively. The behavior seems to be a mixture of bang-bang control and linear control. Around $\theta = 0$, we should notice that a bang-bang control rule was found by the algorithms with $3 \times 3 \times 6 \times 6$ boxes whereas a linear control rule was found by the algorithms with $3 \times 3 \times 5 \times 5$ boxes.

Anyway, the algorithm founds preferable solutions.

We also tried conventional Q-learning using 3x3x10x5 and 3x3x12x6 boxes, which are just twice the boxes. But they could not hold the pole vertical.

5 DISCUSSION

Evaluation: We cannot conclude superiority of the proposed method, because the complexity of our approach is obviously larger than that of conventional Q-learning in the experiment. The experiment shows that the learning of the local linear controllers can make up for lack of learning ability on the high-level module.

Effects of the Partitioning Location: In our test cases, different discretizations led the algorithms to learning quite different control rules, especially around $\theta = 0$. It makes clear that the shape and location of the quantized regions have great effects on the performance. Joint use of variable-grid methods (e.g. partigame [10]), would be needed for generating discrete representation rather than the use of the fixed grids.

Time Intervals of High-level Decision Making: In our approach, time intervals of decision making on the high-level hierarchy are mostly owing to the size of the state boxes. It can be seen as an adaptive choice of time intervals in continuous-time domains [11] [2].

Learning with Extremely Coarse Value Approximation: Many DP-based (RL) algorithms are motivated by a desire for finding highly accurate value functions. However, it costs too much memory to approximate such functions in many cases. In contrast, our approach does not stick to finding accurate value functions, because the policy can be improved by a gradient-ascent search without explicitly computing value estimates.

6 CONCLUSION

This paper presented a hierarchical RL algorithm composed of TD method and local linear controllers to solve a non-linear control problem in which state and action spaces are continuous. It is a hybrid method between DP-based value estimation and policy improvement by gradient-ascent without value estimation. The continuous state-action space is discretized into an array of coarse boxes, and roughly the high-level hierarchy estimates the value functions over the discrete space to find a globally preferable policy. The local linear controllers are to improve the policy by stochastic gradient-ascent. This method does not need to fit accurate value functions, therefore it may be promising to overcome the curse of dimensionality. The algorithm was applied to a simulation of a cart-pole swing-up problem, and better solutions are found than those of traditional discrete RL methods.

REFERENCES

[1] Barto, A. G., Sutton, R. S. & Anderson, C. W.: Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems, *IEEE Transactions on Systems, Man, and Cybernetics*, vol.SMC-13, no.5, pp.834–846 (1983).

[2] Buckland, K. M. & Lawrence, P. D.: Transition Point Dynamic Programming, *Advances in Neural Information Processing Systems 6*, pp.639–646 (1993).

[3] Davies, S.: Multidimensional Triangulation and Interpolation for Reinforcement Learning, *Advances in Neural Information Processing Systems 9*, pp.1005–1011 (1996).

[4] Davies, S., Ng, A. Y. & Moore, A.: Applying Online Search Techniques to Continuous-State Reinforcement Learning, *15th National Conf. on Artificial Intelligence*, pp.753–760 (1998).

[5] Dietterich, T. G.: The MAXQ Method for Hierarchical Reinforcement Learning, *Proceedings of the 15th Int. Conf. on Machine Learning*, pp.118–126 (1998).

[6] Doya, K.: Efficient Nonlinear Control with Actor-Tutor Architecture, *Advances in Neural Information Processing Systems 9*, pp.1012–1018 (1996).

[7] Gullapalli, V.: Reinforcement Learning and Its Application to Control, *PhD Thesis*, University of Massachusetts, Amherst, COINS Technical Report 92–10 (1992).

[8] Kimura, H. & Kobayashi, S.: An Analysis of Actor/Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imperfect Value Function, *15th International Conference on Machine Learning*, pp.278–286 (1998).

[9] Lin, C. J. & Lin, C. T.: Reinforcement Learning for An ART-Based Fuzzy Adaptive Learning Control Network, *IEEE Transactions on Neural Networks*, Vol.7, No.3, pp.709-731 (1996).

[10] Moore A. W. & Atkeson, C. G.: The Parti-game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-spaces, *Machine Learning 21*, pp.199-233 (1995).

[11] Pareigis, S.: Adaptive choice of grid and time in reinforcement learning, *Advances in Neural Information Processing Systems 10*, pp.1036–1042 (1998).

[12] Parr, R. & Russell, S.: Reinforcement Learning with Hierarchies of Machines, *Advances in Neural Information Processing Systems 10*, pp.1043–1049 (1998).

[13] Sutton, R. S. & Barto, A.: Reinforcement Learning: An Introduction, MIT Press (1998).

[14] Sutton, R. S., Precup, D. & Singh, S.: Intra-Option Learning about Temporary Abstract Actions, *Proceedings of the 15th International Conference on Machine Learning*, pp.556–564 (1998).

[15] Tsitsiklis, J. N., & Roy, B. V.: An Analysis of Temporal-Difference Learning with Function Approximation, *IEEE Transactions on Automatic Control*, Vol.42, No.5, pp.674–690 (1997).

[16] Watkins, C. J. C. H., & Dayan, P.: Technical Note: *Q*-Learning, *Machine Learning 8*,pp. 55–68 (1992).

[17] Williams, R. J.: Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning, *Machine Learning 8*, pp.229–256 (1992).