Exploiting Value Statistics for Similar Continuing Tasks

Fumihide Tanaka and Masayuki Yamamura
Dept. of Computational Intelligence and Systems Science, Tokyo Institute of Technology
4259, Nagatsuta-chou, Midori-ku, Yokohama, 226-0026, Japan
boom@es.dis.titech.ac.jp

***Abstract*— In this paper, we try to consider interaction design for adaptation from the viewpoint of transfer of knowledge. Recent advancements in robotics are amazing, and their interaction processes with outside world (including human) are getting to be longer in time scale. We will investigate these matters in an abstract agent that faces multiple learning tasks within its lifetime, transferring past learning experiences to improve its performance. We formulize the multitask reinforcement learning problem at first, and then we present two ways of incorporating past learning experiences into the agent's learning algorithm.**

I. Introduction

We live in the era that autonomous robots will soon be with our daily lives. These robots interact with their outside world based on wide range of technologies aimed for highly embedded systems. As one direction among them, we focus on a lifelong property in the autonomous robots. In other words, we consider the agent that deals with multiple learning tasks within its lifetime exploiting past learning experiences to improve its performance. To begin with, let us present one fundamental example here:

Imagine there is a cleaning robot working at a hotel. Its job is to clean up a room every day after the customer checks out. (For simplicity, here we consider that only one room is done in a day by one robot) The structures of all rooms are basically the same, but sometimes there are cases where customers move desks or chairs by their own, and then the environment for the robot isn't totally static. Under these settings, the robot has to work every day through years, finishing the job as fast as possible for preparing another customer's check-in.

Now, if we view that each day's job is one task for the robot to learn how to clean up the hotel's room, then there given multiple tasks through years. And it is essential for the robot to keep its past learning experiences and utilize them in the future learning tasks to solve them faster. Our motivation of this study rises here, and we address one way of developing software technologies about reinforcement learning for robots that have long-term lifetime like human beings. Reinforcement learning[7] is a vigorously studied area in machine learning researches. It is a general framework of learning through trial and error and it can model wide range of interaction processes. In contrast to the conventional techniques dealing with single task learning, we consider Multitask Reinforcement Learning towards the global end mentioned above. Generally, in reinforcement learning, there need a great number of learning trials, and then if we consider applying it in the multitask domain, exploiting past learning experiences becomes crucial as it would have the possibility to reduce the number dramatically.

Based on foundations explained in Section II, discussions begin with how to formulize the learning problem formally (Section III), which has been difficult in these areas. The central idea here is to consider the distribution of tasks, each of which is defined as a Markov Decision Process. We can set environmental features that are common to all tasks by using value statistics introduced here. Multiple tasks are sampled from the distribution and presented to the agent one by one through its lifetime. The agent's objective is to maximize the total reward through the lifetime as well as the conventional return in each task. To do this, it has to be endowed an important ability to improve its reinforcement learning performance by using past learning experiences that were obtained as value statistics.

How to exploit the value statistics facing a new task is another topic of this paper. We propose two methods that are independent but helped each other: One is to exploit them in Prioritized Sweeping (Section IV), a general standard of determining the order of simulated learning. By using deviation information about each value, reliability difference in initial values of learning is considered, leading to more accurate calculation of the order (priority in simulated learning). The other is to exploit them in Directed Exploration (Section V), which guides the agent's action selection strategy as biased ways. This method offers the agent not only effective exploration, building the model, but also further reliable calculation of the priority explained above. All these methods are tested and evaluated in computer simulation experiments showing their effectiveness.

II. Foundations [7]

A. Reinforcement Learning in MDP

There is a learning agent that interacts with its outside (environment) at some discrete time scale. At each time step, the agent selects an action $a_t \in \mathcal{A}$ based on its observation of the environment's state $s_t \in \mathcal{S}$. In response to the a_t, the environment returns a numerical reward r_{t+1} and the next state s_{t+1} to the agent as a consequence of a state-transition. Basically, the environmental dynamics is modeled through a finite Markov Decision Process (MDP). A particular finite MDP is defined by four sets $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}>$, where $\mathcal{P}$ is a set of transition probabilities $P^a_{ss'} = \Pr\{s_{t+1} = s' \mid s, a\}$, and $\mathcal{R}$ is a set of reward functions $R^a_{ss'} = E\{r_{t+1} \mid s, a, s'\}$. The agent's objective is to find optimal policy $\pi(s, a)$, which is a mapping from states to probabilities of taking each possible action. It is updated through learning to maximize the expected discounted future reward from each state s:

$$v^{\pi}(s)=E\left\{ r_t+\gamma r_{t+1}+\gamma^2 r_{t+2}\cdots \middle|\, s_t=s,\pi\right\} \tag{1}$$

where γ is a discount rate. This $v^{\pi}(s)$ specifies the value of a state s under policy π, and v^{π} is called the value function for policy π. Based on this,

$$v^*(s)=\max_{\pi} v^{\pi}(s) \tag{2}$$

can be defined, and v^* is called the optimal value function. By using this, delayed rewards can be dealt with, and this is one feature of RL. These value functions can be estimated through trial and error. That is, the agent can learn them directly from raw experience without a model of the environmental dynamics. Various approaches were proposed to estimate them, and we will later use Q-learning[12] among them. In Q-learning, the action-value $q(s,a)$ is used instead of $v(s)$. This method is known to be able to estimate the optimal value function under some assumptions, and is widely used in the literature of RL fields.

B. Reinforcement Learning with Models

Usually, in reinforcement learning, there need a large number of learning trials to estimate the optimal value function. One general way to reduce this cost is introducing a model, which is often called as model-based reinforcement learning [8]. In addition to the normal learning process that is done by real experiences, a model of the environment and simulated experiences (planning process) in it are used to estimate the values, and then its learning speed is accelerated.

Basically, simulated experiences are obtained by random backups (updating values). But it is well known that the order of backups is crucial for the better performance in model-based reinforcement learning. Prioritized Sweeping [4], [6] decides the order by using a priority metric p below:

$$p=\left|\; q^T-q^C\;\right| \tag{3}$$

where $q^C=q(s,a)$ and q^T is various by the difference of algorithms. For example, when we use Q-learning, $q^T=r+\gamma\max_{a'} q(s',a')$, or when we use SARSA[7], $q^T=r+\gamma q(s',a')$. In Prioritized Sweeping, backups having larger p are prioritized in its order.

III. The MTRL Problem

A. A Task and the Distribution of Tasks

From now on, we define a task by MDP and its running time τ, that is, by specifying five sets $<\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\tau>$. As we reviewed in Section II-A, a set of the optimal values $v^*(s)$ can be determined if we specify the MDP. Therefore, we can say now that every task has its own optimal value set $\{v^*(s)\}$ respectively. Then, we consider the distribution of the tasks. In other words, we define a class over multiple MDPs from which we can sample tasks independently. To describe the property of the class, we use a set of $V^*(s)$ that denotes the distribution of $v^*(s)$. Specifying $V^*(s)$ can be viewed as setting a relationship between multiple tasks. We can represent environmental features by using it, and we name the class as MDPs with Biased-Values (BV-MDPs). In this paper, we presuppose that each $V^*(s)$ is independent for different s.

B. The Total Reward through a Lifetime

As we mentioned in Section II-A, the objective of the agent in reinforcement learning is to maximize the expected return in each state. In addition to this, we consider here the total reward through the agent's lifetime. Formally, it is described as follows:

$$TR=\sum_{i=1}^{N}\int_{t_{i-1}}^{t_{i-1}+\tau_i} r_t\,dt \quad \left(t_i=\sum_{j=1}^{i}\tau_j,\ t_0=0\right) \tag{4}$$

where N is the total number of tasks, and τ_i is the running time of task i. We call this TR as the total reward through a lifetime.

To solve the equation (4) directly is pretty hard and currently impossible. Then, instead of it, we aim for maximizing the sub-total reward below in every task by exploiting past learning experiences obtained by then. This is valid because all tasks are sampled independently in our problem setting.

$$STR=\int_{t_{i-1}}^{t_{i-1}+\tau_i} r_t\,dt \quad \left(t_i=\sum_{j=1}^{i}\tau_j,\ t_0=0\right) \tag{5}$$

C. Maintaining Value Statistics

In reinforcement learning, the most general data as learning results are estimated values. As we explained in Section III-A, there are relationships between tasks sampled from BV-MDPs in their optimal values. In this section, we consider extracting the relationships from value statistics. We view them as learning experiences, and let the agent maintain them through tasks. Value statistics are, as it were, the past learning experiences that are compressed. In our problem domain, it is inappropriate to store values of all tasks because the number of them is large and the memory costs grow up enormously. Therefore, such a method as a modular-approach about value functions is not suitable. We need here more compact representation, and hire statistics. After finishing each task, the agent can easily update the mean and deviation about its action values (for all s,a):

$$\overline{Q}(s,a)=\frac{1}{n}\sum_{i=1}^{n} q(s,a)_i \tag{6}$$

$$s_{Q(s,a)}=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\,q(s,a)_i-\overline{Q}(s,a)\,\right)^2} \tag{7}$$

where i indicates a task index, and n is the number of tasks that were previously solved. These quantities represent features in the environment that are common to all of the tasks. They can also be viewed as that if $\overline{Q}(s,a)$ is used as an initial value in a new task, then its reliability is given by $s_{Q(s,a)}$.

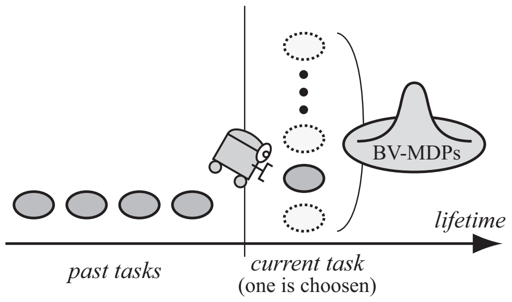

Fig. 1. The MTRL agent

D. Formalizing the Problem

Based on arguments so far, in this section we formulize the MTRL problem. We consider an agent that faces multiple learning tasks one by one through its lifetime, utilizing past experiences for improving its current and future learning performance (Figure 1). The agent's objective is to maximize its total reward TR, in addition to the conventional return in each task. A task is defined as an independent sample from BV-MDPs, and each is presented to the agent in every τ_i time until the end of its lifetime.

IV. Prioritized Sweeping with Value Statistics

The ability to transfer past learning experiences and utilize them for improving future learning performance is at the core of the MTRL agent. In the following two sections, we will discuss how they are realized by using value statistics introduced in Section III-C. First, in this section we will consider incorporating them into Prioritized Sweeping, a fundamental procedure in model-based reinforcement learning.

A. Exploiting Value Statistics in Model-Based Reinforcement Learning

To maximize (5), the agent has to estimate the optimal value function as fast as possible in each task. In the situation where the learning speed is crucial, usually a model-based approach explained in Section II-B is used in reinforcement learning. Now we explain how previously obtained value statistics are combined with the model-based reinforcement learning.

1) Exploiting $\overline{Q}$: Usually in reinforcement learning, initial values (values at the beginning of learning process) are set randomly or zero, and there isn't specific standard that is considered to be the best though they affect the learning-speed very much. (A few exceptions are in optimistic initial values [9], [7] and an analysis [3])

In the MTRL problem, however, we can employ $\overline{Q}(s,a)$ obtained by then as initial values. The more static the environmental dynamics is, the more effective it would be.

$$q_0(s,a) = \overline{Q}(s,a),\ for\ all\ s,a \tag{8}$$

2) Exploiting s_Q: To make model-based reinforcement learning effective, there are two precepts in its ways of simulated backups. As the number of them is fixed at X times per one (real) action, it is important to let one simulated backup *as large as* and *as precise as* possible. Prioritized Sweeping (PS) explained in Section II-B was designed to realize the former aim that is about the size of each backup. Now, we consider the latter one that is about accuracy of each backup. As we mentioned in Section III-C, each $\overline{Q}(s,a)$ has its own reliability represented by $s_{Q(s,a)}$. We make use of this feature. Most reinforcement learning methods like Q-learning are what we call *bootstrapping* one: each value is estimated by using other values that themselves are also estimates. Therefore, in the middle of learning phase, the values are not always accurate, that is, there often happen cases where they are far from the optimal values. Now, it is expected that we can use the deviation s_Q to decrease the number of wrong backups caused by the inaccurate values. This idea is realized by introducing a new priority metric designed for the MTRL problem:

$$\begin{aligned} p_E &= E\left[\,\left(Q^T - Q^C\right)^2\right] \\ &\simeq \left(\overline{Q^T} - \overline{Q^C}\right)^2 + \left(s_{Q^T}\right)^2 + \left(s_{Q^C}\right)^2 \end{aligned} \tag{9}$$

The squared procedure is valid as the priority metric is originally positive (cf. (3)) and only the relative size is concerned. By using this p_E, the deviation information about each $q(s,a)$ is properly exploited, and more precise value estimation can be realized.

Value statistics are updated at the end of each task's learning phase. In contrast to the mean information $\overline{Q}$ that is used only when each task starts, the deviation s_Q has been hired through the task all the way. Value statistics are *universal* information in that they are obtained through past tasks. Then, if one wants to focus on the performance of convergence within a task, the information should be shifted gradually toward more *local* type. To do this, we prepare another deviation $s'_Q(s,a)$ that represents the fluctuation of $q(s,a)$ within a task updated in every step of learning. $s'_Q(s,a)$ is set to 0.0 at the beginning of each task for all s,a. Then, the weighted sum of $s_Q(s,a)$ and $s'_Q(s,a)$ is used changing the weights. By using this, the deviation is shifted gradually toward the *local* type. Currently, a slope parameter β that indicates a point where the weight (rate) becomes 0.0 is determined by trial and error in computer simulation. Detailed description here can be found in [11].

B. Experiments

1) Objective: In this section we set several examples of the MTRL problem explained previously by using computer simulation. We conduct experiments there and evaluate the effectiveness of our approach. For the clear understanding about the results, there are some remarks:

- We let the agent's lifetime enough long for acquiring accurate statistics. Then, by using new tasks, we compare results by our method that exploits value statistics and those by simple PS. This is based on the argument in Section III-B.
- The purpose of the agent in reinforcement learning is to maximize some criterions about reward. When we evaluate its performance, such a standard like an average reward is sometimes inappropriate as it is affected by the differences of the agent's policy (action selection).

Then, we evaluate methods by measuring accuracy of value estimation. We calculate the mean-squared error (MSE) between estimated values and the optimal values to judge the accuracy. As we explained in Section II-A, the value of a state is a metric that generalizes acquiring rewards, and estimating it faster leads to getting more rewards in τ period.

2) Plan: Experiments are conducted on the square random gridworld (SRG) illustrated in Figure 2. The gridworld is a straight framing of MDP and it has been commonly used especially in model-based reinforcement learning. Each cell of the grid corresponds a state, and four actions are possible there: north, south, east, and west, which stochastically cause the agent to move one cell in the corresponding direction on the grid. The transition probabilities are determined when a task is given to the agent, which will be explained in detail later. The agent cannot move toward outside. After reaching the goal state (G), the agent gets a goal reward and returns to the start state (S) to begin a new episode. This problem can be represented by MDP, and there the agent aims for estimating the optimal values through trial and error.

Consider a situation where multiple tasks (MDPs) are given to the agent one by one. When a new task (MDP) is given, each transition probability is sampled from a normal distribution. Here, we form BV-MDPs by $< \mathcal{S}, \mathcal{A}, \mathcal{DP}, \mathcal{R} >$ where $\mathcal{DP}$ indicates the set of the distribution. Each sampled probability $P^a_{ss'}$ $(s \neq s')$ represents the probability of success in corresponding state-transition. In other case, the state is unchanged. That is, $P^a_{ss} = 1.0 - P^a_{ss'}$.

In each experiment under common BV-MDPs, at first 100 tasks are sampled to calculate value statistics. Then, 10 new are sampled as well to investigate the effects of proposed methods. Results are evaluated by the averaged mean-squared error (averaged MSE) between $q(s,a)$ and $q^*(s,a)$ for all s, a over the 10 tasks. Regarding the agent's action selection (exploration strategy), we use ϵ-$greedy$ selection where a random action is selected with probability ϵ, and in other cases an action whose value is the highest is selected [7]. Basically, we compare three kinds of methods in all experiments: simple Prioritized Sweeping (PS), PS with $\overline{Q}$ as initial values, and PS with both value statistics ($\overline{Q}$ and s_Q). The learning rate is set to 0.01, ϵ is 0.5 and the discount rate to 0.95. Under these settings, we conduct two kinds of experiments below:

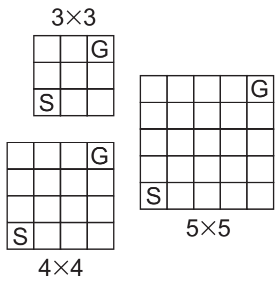

Fig. 2. The square random gridworld

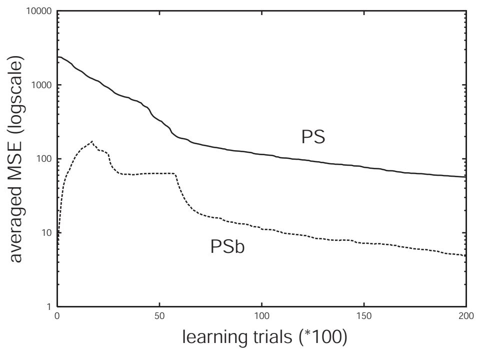

Fig. 3. Learning curves for PS and PSb obtained by experiments conducted in SRGs whose size is 10×10. The performance measure shown is MSE between the value function learned and the true value function that is averaged over 10 tasks. The vertical axis is log scale.

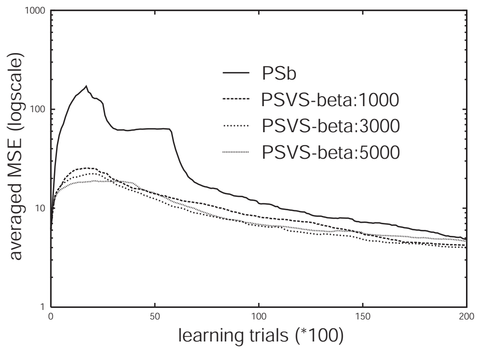

Fig. 4. Learning curves for PSb and PSVS obtained by experiments conducted in SRGs whose size is 10×10. Three lines are plotted in case of PSVS by the differences of β.

[Experiment-A:] We compare results by using SRG whose size is 10×10. 110 tasks are sampled and three methods above are evaluated under them. State-transitions are totally stochastic, and all $P^a_{ss'}$ $(s \neq s')$ are sampled from the normal distribution whose mean is 0.5 and variance 0.02.

[Experiment-B:] This time, the SRG size is fixed to 10×10. 9 environments are prepared by the difference of the number of states whose transitions are stochastic (non-deterministic): $10, 20, \ldots 90$. Regarding the states, transition probabilities are sampled from the normal distribution that is the same as Experiment-A.

3) Results: Experiment-A: Figure 3 shows a comparison result between the simple prioritized sweeping (PS) and it exploiting $\overline{Q}$ in its initial values (PSb: See Section IV-A.1). It describes the effects of introducing $\overline{Q}$ as initial biases. It focus on the beginning stage of learning phase as a typical character introducing them can be seen here: at first the performance of PSb gets wrong temporary, and then it improves all the way. If the experiment continues longer, then the result of PSb will be better than the start point. There is an offset between two lines plotted and it means the performance gain of introducing $\overline{Q}$. The tendency is common to all experiments conducted here.

Next, Figure 4 shows another comparison result between PSb and PS exploiting both $\overline{Q}$ and s_Q (PSVS). In case of the latter, three lines are plotted by the differences of the

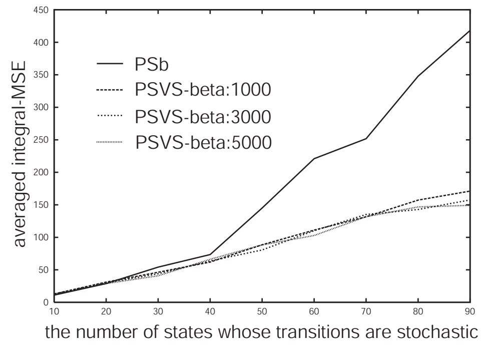

Fig. 5. Summary of Experiment-B. Experiments are conducted in SRGs whose size is 10×10. We can see the performance difference between PSb and PSVS with the number of states whose transitions are stochastic increasing.

parameter β (1000, 3000, 5000). It describes the effects of introducing s_Q in calculating priority of simulated backups (See Section IV-A.2). We can see here that the performance of the beginning stage of learning phase is improved. This is because the number of mis-backups is decreased by considering deviation information of q value when calculating the priority. It realizes more precise estimation of values in model-based reinforcement learning.

4) Results: Experiment-B: Figure 5 summarizes the results. The horizontal axis means the number of states whose transitions are stochastic. The vertical axis means integral-MSE until 1000 steps that are averaged over 10 experiments. We can find in them that the performance difference between two methods widens as the value of the horizontal axis increases. In case of 10, there seems no difference (PSVS takes more computational costs in exploiting s_Q), and it grows up gradually. After all, the effects of introducing $\overline{Q}$ is stronger in the static environment, and adding s_Q soften its performance decay when the environment becomes more stochastic.

V. Directed Exploration with Value Statistics

So far we didn't care so much about the agent's action selection (exploration) strategy. But it is well known that directed exploration techniques that direct the action selection behavior to the most interesting parts of the state-action space can reduce the learning time significantly. Thrun [10] proved that in some domains, reinforcement learning using directed exploration could be performed in polynomial time though in the undirected case it was expected to scale exponentially with the size of the state-action space grown up [13]. Up to now, many kinds of these techniques have been proposed [10], [2], [1], [14], [5] though they are within a single task learning domain. In this section, we consider them in the MTRL problem by using value statistics.

A. Exploiting Value Statistics in Directed Exploration

In the MTRL problem, we can utilize value statistics ($\overline{Q}$ and s_Q) facing a new task. As we explained in Section III-C, s_Q can be viewed as representing reliability of $\overline{Q}$ used in initial values. Then, it is natural that exploration should be directed in such a way that low reliability areas are searched more compared with others.

Further, there is another significant point here. If exploration is done in the undirected manner, sometimes it would happen cases where values that have already been estimated enough accurately are disturbed by unexpected random backups caused by next states' values that are inaccurate. In the experiments of previous section, we found that introducing $\overline{Q}$ caused temporary declining performance though it was softened by introducing s_Q. One of the sources of this is thought to be the undirected exploration. Now we can expect more improved performance by using directed exploration.

In this section, we will present several ways of incorporating value statistics into conventional exploration techniques, ϵ-$greedy$ and $softmax$ action selection. Then we will investigate their effectiveness by comparing them with the previous results that were obtained by using undirected exploration.

1) ϵ-$greedy$ method with value statistics: As we explained in Section IV-B.2, ϵ-$greedy$ action selection is such a method that a random action is selected with probability ϵ, otherwise it that has the largest value is selected. Now, we can modify it in two ways:

prob. ϵ :	Select an action whose $s_{Q(s,a)}$ is the largest
$1.0-\epsilon$:	Select an action whose $Q(s,a)$ is the largest

This method (Type1) omits random exploration. Exploration is done in order of deviation information. If $s_{Q(s,a)}$ isn't given appropriately, it would be in danger of falling deadlock.

prob. ϵ :	Select an action randomly
$1.0-\epsilon$:	Select an action whose $s_{Q(s,a)}$ is the largest

This method (Type2) intermixes random exploration with directed one using $s_{Q(s,a)}$. It also can be viewed as an action selection strategy that is *greedy with* $s_{Q(s,a)}$.

2) Softmax method with value statistics: Regarding the action selection of the agent in reinforcement learning, there has been another widely used method called *softmax* action selection [7]. It varies the action probabilities as a graded function of estimated value. To incorporate value statistics into this method, now we present a new action probability below:

$$\frac{e^{s_{Q(s,a)}/T}}{\sum_{for\ all\ a} e^{s_{Q(s,a)}/T}} \tag{10}$$

where T is a positive parameter called the temperature. High temperatures promote randomness, and low them promote greediness. This method is somewhat resemble to the Type2 method presented in Section V-A.1. It realizes another way of directing randomness in action selection. The greedy action with $s_{Q(s,a)}$ is given the highest selection probability, though all the others are ranked according to their $s_{Q(s,a)}$ respectively. It is difficult to judge whether softmax method or ϵ-$greedy$ method is superior [7]. The situation is unchanged also in our case, which will be shown in the next section.

B. Experiments

1) Objective: In this section, we will investigate our methods presented in Section V-A. Our main interest is to know how much degree the performance of PSVS (Prioritized Sweeping with Value Statistics: Section IV) is improved by introducing DEVS (Directed Exploration with Value Statistics:

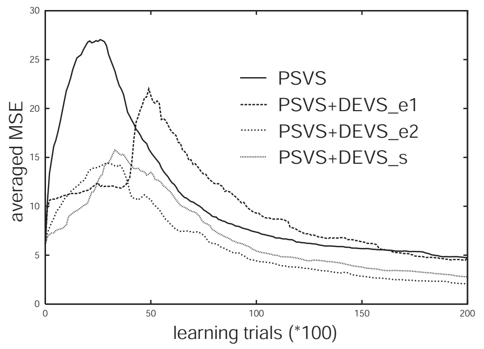

Fig. 6. Performance comparison between four methods. Experiments were done in SRGs whose size is 10×10. Results were averaged over 10 tasks. (Parameter settings: $\epsilon = 0.5$, $T = 0.01$, $\beta = 3000$)

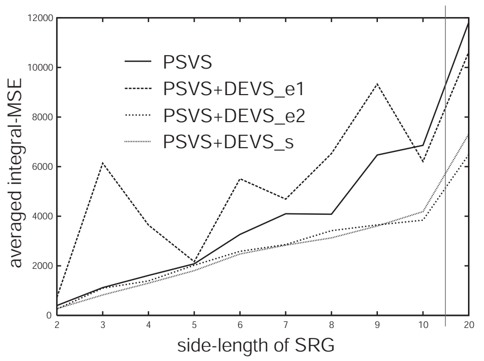

Fig. 7. Performance comparison between four methods. The horizontal axis indicates the size of SRG. The vertical axis plots integral-MSE until 10000 steps that is averaged over 5 overall experiments.

This section's topic). Therefore, we will compare all results with those presented in Section IV.

2) Plan: Regarding the experimental environment, we use the square random gridworld (SRG) that is the same as experiments in Section IV. In Section V-A, we presented three methods concerning with DEVS:

- ϵ-$greedy$ method with value statistics, Type1 (DEVS_e1)
- ϵ-$greedy$ method with value statistics, Type2 (DEVS_e2)
- Softmax method with value statistics (DEVS_s)

We compare results of them with each other and those by the simple PSVS that uses the standard ϵ-$greedy$ method in its action selection.

3) Results: Figure 6 illustrates a comparison of learning curves by four methods (PSVS, PSVS+DEVS_e1, PSVS+DEVS_e2, PSVS+DEVS_s) in SRG whose size is 10. Figure 7 shows another comparison that is done by experiments under different sizes of SRGs. We can see there that the result of PSVS_e1 is quite instable, and it is inferior to simple PSVS.

VI. Conclusion

Along with recent advances in autonomous agents and robotics, we aim for scaling up the basic framework of reinforcement learning towards multitask-oriented. Our motivation arises from consideration that interaction process can be viewed as tasks, and it is getting to be longer in time scale. We presuppose such an application in our mind as a cleaning robot that works everyday at hotels. If we consider each day's job as one learning task, then there continue multiple tasks given to the robot through a year, and it is crucial for the robot to maintain and exploit past learning experiences to improve its daily performance.

Based on the motivation, we set two objectives. One was to formulize the reinforcement learning problem dealing with multiple learning tasks. The other was to present some concrete learning algorithms that work effectively.

For the first objective, we defined the MTRL (Multitask Reinforcement Learning) problem by introducing an environmental class called BV-MDPs that specified the distribution of tasks. By using the class, we could set environmental features that were common to all tasks. For the second objective, we proposed two independent ways incorporating value statistics into reinforcement learning. One was incorporating them with Prioritized Sweeping that realized the better performance in model-based reinforcement learning. The other was with Directed Exploration that realized not only effective searches but also reduction of unexpected harmful backups. We presented three specific methods based on ϵ-$greedy$ and $softmax$ action selection. All these ways were tested in computer simulation experiments and we evaluated the effectiveness in detail.

References

[1] Dayan, P. and Sejnowski, T. J. "Exploration Bonuses and Dual Control", *Machine Learning*, Vol.25 No.1 pp.5–22, 1996.

[2] Kaelbling, L. P. *Learning in Embedded Systems*, MIT Press, 1993.

[3] Koenig, S. and Simmons, R. G. "The Effect of Representation and Knowledge on Goal-Directed Exploration with Reinforcement Learning Algorithms", *Machine Learning*, Vol.22 No.1 pp.227–250, 1996.

[4] Moore, A. W. and Atkeson, C. G. "Prioritized sweeping: Reinforcement learning with less data and less real time", *Machine Learning*, Vol.13 pp.103–130, 1993.

[5] Meuleau, N. and Bourgine, P. "Exploration of Multi-State Environments: Local Measures and Back-Propagation of Uncertainty", *Machine Learning*, Vol.35 No.2 pp.117–154, 1999.

[6] Peng, J. and Williams, R. J. "Efficient learning and planning within the Dyna framework", *Adaptive Behavior*, Vol.1 No.4 pp.437–454, 1993.

[7] Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[8] Sutton, R. S. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming", *Proceedings of the Seventh International Conference on Machine Learning*, pp.216–224, 1990.

[9] Sutton, R. S. "Generalization in reinforcement learning: Successful examples using sparse coarse coding", *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp.1038–1044, 1996.

[10] Thrun, S. "Efficient Exploration in Reinforcement Learning", Technical Report CMU-CS-92-102, Carnegie Mellon University, Computer Science Department, 1992.

[11] Tanaka, F. and Yamamura, M. "Multitask Reinforcement Learning on the Distribution of MDPs", *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2003.

[12] Watkins, C. J. C. H. and Dayan, P. "Q-learning", *Machine Learning*, Vol.8 pp.279–292, 1992.

[13] Whitehead, S. D. "A complexity analysis of cooperative mechanisms in reinforcement learning", *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp.607–613, 1991.

[14] Wiering, M. and Schmidhuber, J. "Efficient Model-Based Exploration", *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 6*, pp.223–228, 1998.