Moratorium Effect on Estimation Values in Simple Reinforcement Learning

Akira Notsu[*1], Yuki Tezuka[2], Katsuhiro Honda[3]

Department of Computer Science and Intelligent Systems, Osaka Prefecture University/1-1
Gakuen-cho, Nakaku, Sakai, Osaka 599-8531, Japan

[*1]*notsu@cs.osakafu-u.ac.jp; [2]tezuka@hi.cs.osakafu-u.ac.jp; [3]honda@cs.osakafu-u.ac.jp*

***Abstract*- In this paper, we introduce low-priority cut-in (moratorium) to chain form reinforcement learning, which we proposed as Simple Reinforcement Learning for a reinforcement learning agent that has small memory. In the real world, learning is difficult because there are an infinite number of states and actions that need a large number of stored memory and learning time. To solve the problem, better estimated values are categorized as "GOOD" in the reinforcement learning process. Additionally, the alignment sequence of estimated values is changed because they are regarded as an important sequence themselves. However, the method is heavily affected by the action policy. If an agent tends to search many states, its memory overflows with low-value data. Thus, low-priority cut-in (moratorium) enhances the method in order to solve this problem. We conducted some simulations and observed the influence of our methods. Several simulation results show good influence on learning.**

Keywords- Reinforcement Learning; Q-learning; State-action Set Categorization

I. INTRODUCTION

Recently, the research on methods of expressing a continuous state and action, which is more similar to the real world situation, has progressed in reinforcement learning [1].

Reinforcement learning needs finite Markov decision process with a finite state and action. Discrete actions and states are required for Q-learning [2, 3] and SARSA [4] which are typical reinforced study methods. However, there are an infinite number of states and actions in the real learning; so, the learning agent needs a lot of memory to learn optimum or suboptimal solutions in the (approximated) Markov decision process. Efficient discrimination of the continuous state and action is one of the main topics.

On the other hand, we proposed Simple Reinforcement Learning (SRL) for a reinforcement learning agent that has a small number of stored memory and learning times [5]. In our method, several estimated values are categorized into ``GOOD'' in the reinforcement learning process [6, 7]. Additionally, the alignment sequence of estimated values is rearranged as their priorities represented by the sequences themselves in the process. This method provides us with ``economical learning'' and more memory for smaller discriminations.

In this paper, we call this method Chain Form Reinforcement Learning and introduce a low-priority cut-in (moratorium), which causes delay of a newer allocated situation and action set. This places a moratorium on the decision to formalize its priority effectively and reduce the influence of the agent's policy on learning.

We conducted some simulations and observed the influence of our methods. Simulation results show good influence on 2D goal search learning.

II. REINFORCEMENT LEARNING

In reinforcement learning, agents can be assumed to visit a finite number of states, and when an agent visits a state, a numerical reward will be collected; negative numbers represent punishments [8]. The agents estimate the value of each state-action which is defined by the averaged future reward that can be accumulated by selecting actions from this particular state.

Q-learning is a standard reinforcement learning technique that works by learning an action-value function that gives the expected utility of taking given action *a* in given state *s* and following a fixed policy thereafter. One strength of Q-learning is that it can compare expected utility *Q* of available actions without requiring a model of the environment.

Expected utility *Q* is iteratively updated. For each state *s* from state set *S*, and for each action *a* from action set *A*, we update Q-value *Q*, which depends on current state *s* of the agent, action *a* selected by the agent, next state *s'* of the agent after taking action *a*, and reward *r* received by the agent after the action. The update is calculated with the following expression:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a' \in A(s')} Q(S',a') - Q(s,a)],$$

where α is the learning rate and γ is the discount factor.

After Q-learning has been finished, the optimal policy and the optimal value function have been found without continuously updating the policy during learning.

DOI:10.5963/IJCSAI0303004

Q-learning Process

```
procedure Q (0)-learning
begin
  initialize Q-table Q, ∀s ∈ S , ∀a ∈ A;
  set initial state s_0 and goal state s_n;
  for cycle :=1 to MAXCYCLE do s :=s_0
    while s ≠ s_n do
      a := ActionSelect(Q, s);
      r := GetReward(s, a);
      s' := GotoNextState(s, a);
      Update Q-Value
      s :=s';
    end
  end
end
```

III. SIMPLE REINFORCEMENT LEARNING (CHAIN FORM REINFORCEMENT LEARNING)

Simple Reinforcement Learning was proposed in order to categorize the *Value*(*s*, *a*) into ``GOOD" (or ``NO GOOD") during the learning, which requires little memory in comparison to Q-learning. Excessively precise expected utility *Q* is not always required in the policy for modestly successful action selection.

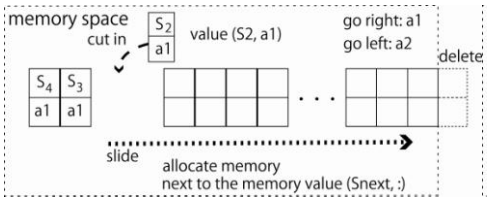


Fig. 1 Newer allocated ``GOOD" state (cut-in)

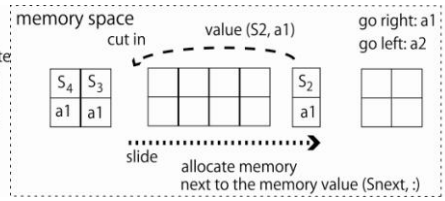


Fig. 2 Reallocated ``GOOD" state (order change)

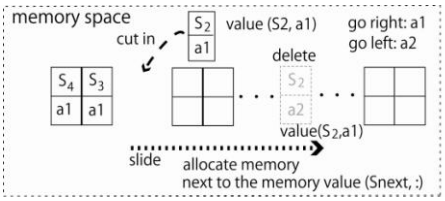


Fig. 3 Reallocated ``GOOD" state (rewrite)

In our SRL process, if *Value(s', :)* is a newer allocated state than *Value(s, :)* and $\max_{a' \in A(s')} Value(s', a)$ == ``GOOD" (the *Value(s, a)* quickly leads to a positive reward), ``GOOD" is applied to *Value(s, a)* and allocated next to *Value(s', :)* (Figures 1-3). In this way, the alignment sequence of estimated values is rearranged **as their priorities**.

SRL Process

```
procedure SRL
begin
  set initial state s_0 and goal state s_n;
  allocate memory Value(s_0, :);
  initialize Value(s_0, :);
  for cycle := 1 to MAXCYCLE do s := s_0
    while s ≠ s_n do
      a := ActionSelect(Value, s);
      r := GetReward(s, a);
      if r == positive and s is a new state,
        allocate memory Value(s, :);
        initialize Value(s, :);
        Value(s, a) := ``GOOD";
      end
      s' := GotoNextState(s, a);
      if Value (s, :) is newer allocated state than Value(s', :) and
          max_{a'∈A(s')} Value (s', a) == ``GOOD"
        reallocated memory Value(s, :) next to the memory Value(s', :) ;
        Value (s, a) := ``GOOD";
        delete inferior state Value(s, :);
      end
      s := s'
    end
  end
end
```


DOI:10.5963/IJCSAI0303004

A. Example of Memory Table

The following examples of memory table are demonstrated. In the example problem, there are five states and two actions. An agent gets rewards 10 (-10) at state s5 (state s1) and goes back to initial state s3, as is shown in Figure 4.

Table 1 shows the Q-table of the example problem after sufficient learning (γ =0.9) and Table 2 shows the Simple Reinforcement Learning Value table of the example problem after sufficient learning.

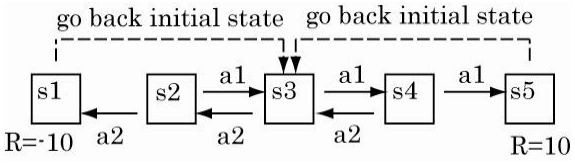


Fig. 4 Example problem

TABLE 1 FINAL Q-TABLE OF EXAMPLE PROBLEM

Q	a1	a2
s1	0	0
s2	8.1	-10
s3	9	7.29
s4	10	8.1
s5	0	0

TABLE 2 IMAGE OF FINAL SRL VALUE-TABLE OF EXAMPLE PROBLEM

Value	a1	a2
s2	GOOD	(unallocated)
s3	GOOD	(unallocated)
s4	GOOD	(unallocated)

B. Low-priority Newer Allocation (moratorium)

SRL is greatly influenced by the agent's policy. If the agent has small memory and has a tendency to select an explorative action, its memory will be filled up with irrelevant data.

In low-priority cut-in (moratorium), a newer allocated situation is made by inserting the situation and action set into the memory with appropriate shift in a deletion direction.

Low-priority cut-in (moratorium) causes delay of a newer allocated situation and action set. This moratorium on decision to formalize its priority reduces the influence of the agent's explorative policy on learning.

We renamed SRL Chain Form Reinforcement Learning (CFRL).

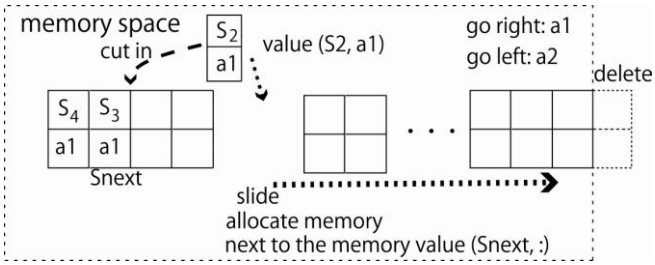


Fig. 5 Low-priority newer allocation (moratorium)

IV. SIMULATION EXPERIMENTS

We simulated a standard game called ``goal search'' as a test case.

A. Goal Search in 2D Cell

In this game, agent actions are up, down, right and left. If the agent goes to the goal, it gets a reward and goes back to the initial state (start state).

The number of states is $n \times n$. *n* is a side size of the field. The agent can observe its state completely.

DOI:10.5963/IJCSAI0303004

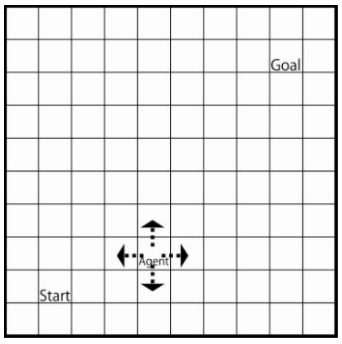


Fig. 6 Goal search in 2D cell

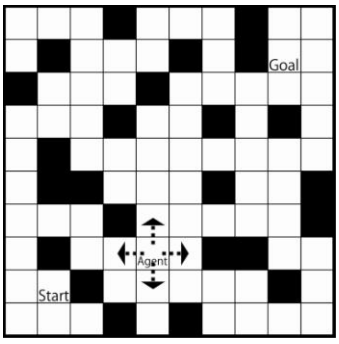


Fig. 7 Example of obstacle

B. *Simulation Parameters*

Agents' policies are ε -greedy. The following parameters are set:

- Learning rate α = 0.1 in Q-learning.
- Discount ratio γ = 0.9 in Q-learning.
- $n \times n = 20 \times 20$ (The number of states in goal search)
- Start state is (2, 2) in 2D-map.
- Goal state is (19, 19) in "D-map.
- Shift in CFRL is 1 (or 2 in tables 4, 5)

Shift in CFRL is 1" means we insert a newer allocated state-action set with one delay. If the agents get a reward (go to goal state) or perform 1,000 actions, their episode ends and the next one begins where agents are set to the initial state.

V. SIMULATION RESULTS

The following figures show learning processes. In addition, goal search examples of agents' actions at the 100th episode are described in Figure 8.

Table 3 shows the necessary amount of memory for the action-state value table.

TABLE 3 AMOUNT OF MEMORY

QL	SRL	SRL64	CFRL64
Sizeof(float) $\times 4 \times 20 \times 20$	2 (bit) $\times 20 \times 20$	2 (bit) $\times 64$	2 (bit) $\times 64$

A. *Goal Search*

Goal search (in Figure 9) results are indicated in Figures 10-15. The vertical axis shows the goal turn, and the horizontal axis shows the number of episodes. The blue, green, red and cyan lines correspond to Q-learning agent, SRL agent, SRL64 agent that has only 64 states of memory and CFRL64 agent that has only 64 states of memory, respectively. Chain Form Reinforcement Learning (CFRL) indicates an SRL with a low-priority newer allocation agent.

The mean and the median of the number of learned actions during 500 episodes are shown in Table 4.

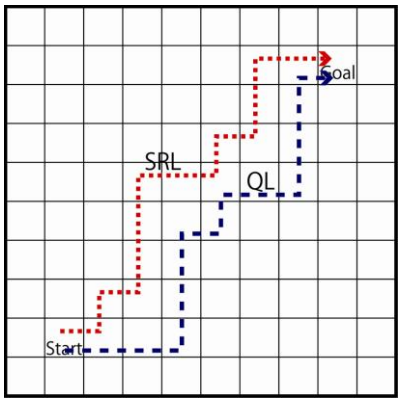


Fig. 8 Examples of learned actions at 100th episode

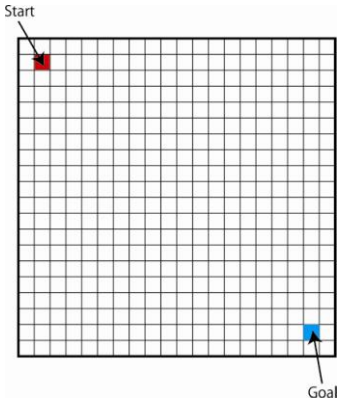


Fig. 9 2D map (map1)

DOI:10.5963/IJCSAI0303004

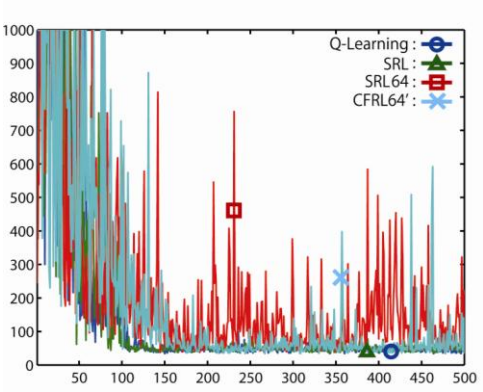


Fig. 10 Example of learning process ($\mathcal{E}$ = 0.2, map1)

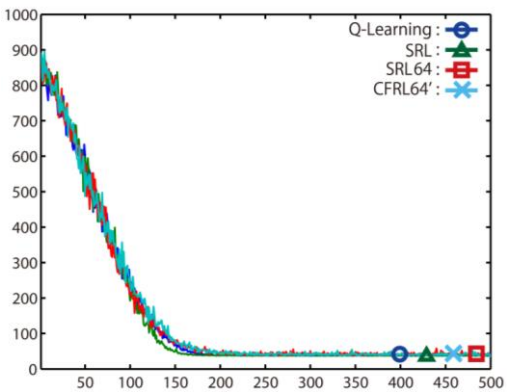


Fig. 11 Mean learning process ($\mathcal{E}$ = 0.05, map1)

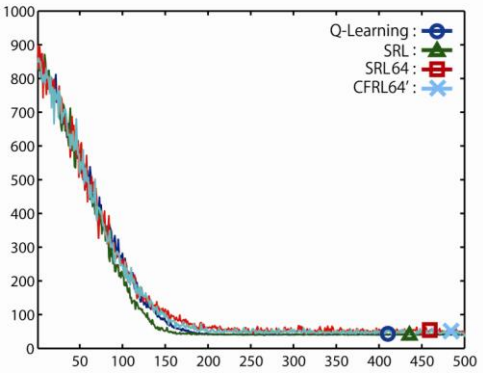


Fig. 12 Mean learning process ($\mathcal{E}$ = 0.1, map1)

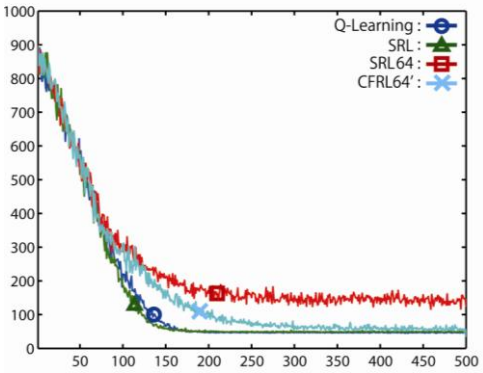


Fig. 13 Mean learning process ($\mathcal{E}$ = 0.2, map1)

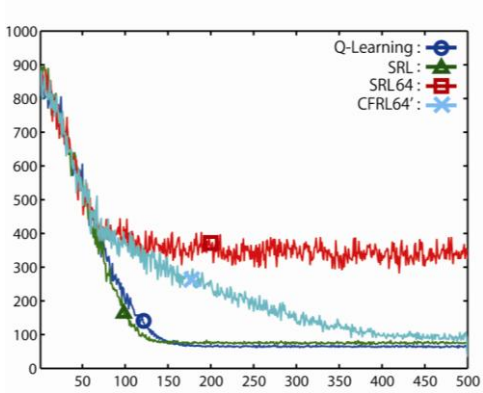


Fig. 14 Mean learning process ($\mathcal{E}$ = 0.4, map1)

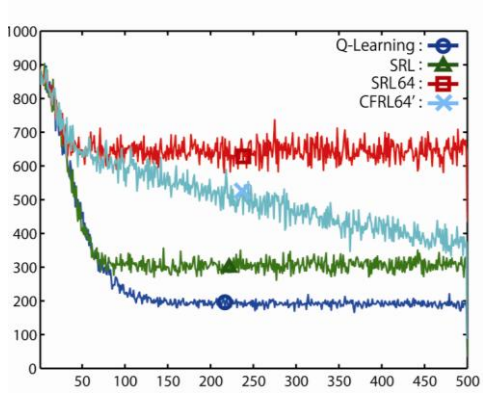


Fig. 15 Mean learning process ($\mathcal{E}$ = 0.8, map1)

TABLE 4 MEAN (AND MEDIAN) OF THE NUMBER OF LEARNED ACTIONS DURING 500 EPISODES (100 AGENTS, MAP1)

$\mathcal{E}$	QL	SRL	SRL64	CFRL64 (Shift = 1)	CFRL64 (Shift = 2)
0.05	34.82 (34)	34.82 (34)	34.74 (34)	34.36 (34)	34.64 (34)
0.1	34.70 (34)	34.60 (34)	35.04 (34)	34.22 (34)	34.38 (34)
0.2	34.24 (34)	34.34 (34)	123.84 (88)	34.16 (34)	34.18 (34)
0.4	34.06 (34)	34.32 (36)	326.78 (329)	34.82 (41)	34.30 (34)
0.8	34.00 (34)	36.40 (36)	433.96 (354)	87.44 (66)	36.42 (36)

B. Goal Search with Obstacles

Goal search with obstacles (in Figure 16) results are described in Figures 17-22. The vertical axis shows goal turn and the horizontal axis shows the number of episodes. The blue, green, red and cyan lines correspond to Q-learning agent, SRL agent, SRL64 agent that has only 64 states of memory and CFRL64 agent that has only 64 states of memory, respectively.

The mean and the median of the number of learned actions during 500 episodes are shown in Table 5

DOI:10.5963/IJCSAI0303004

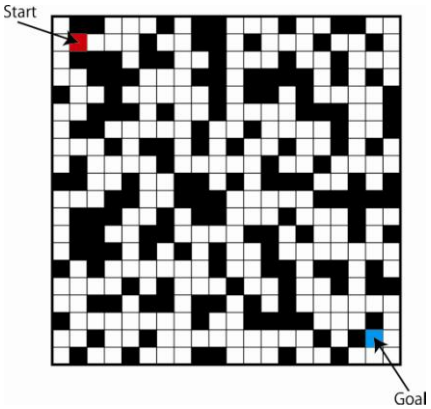


Fig. 16 2D map with obstacle (map2)

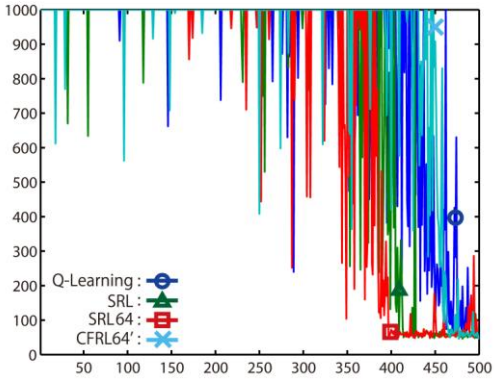


Fig. 17 Example of learning process ($\mathcal{E}$ = 0.2, map2)

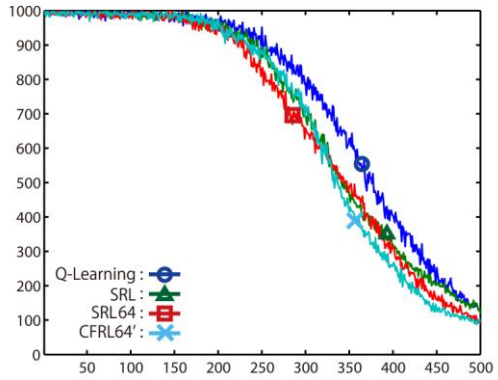


Fig. 18 Mean learning process ($\mathcal{E}$ = 0.05, map2)

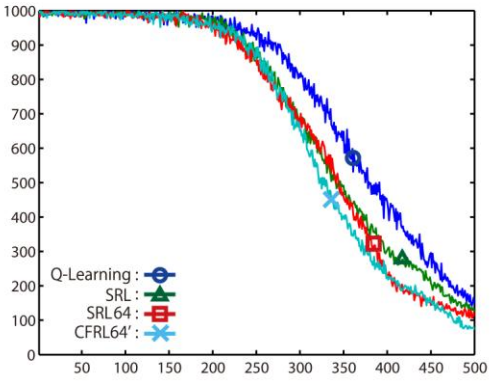


Fig. 19 Mean learning process ($\mathcal{E}$ = 0.1, map2)

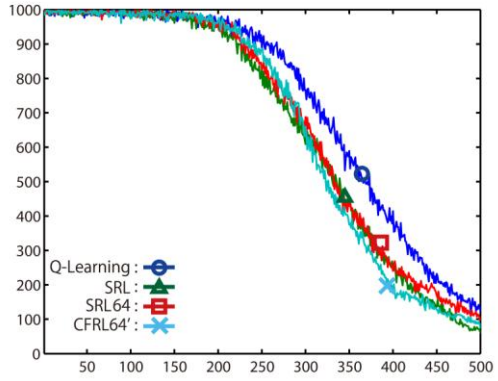


Fig. 20 Mean learning process ($\mathcal{E}$ = 0.2, map2)

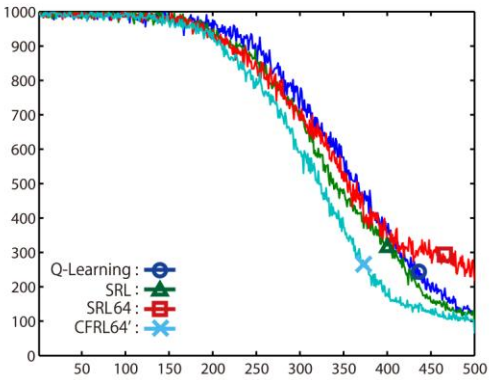


Fig. 21 Mean learning process ($\mathcal{E}$ = 0.4, map2)

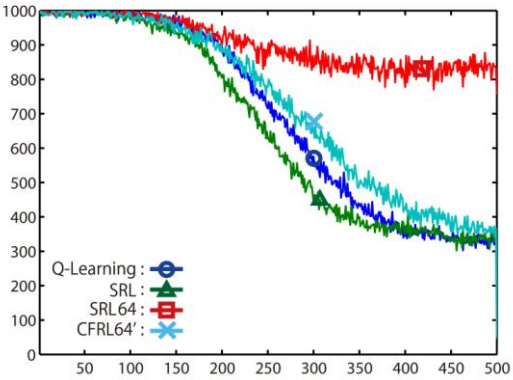


Fig. 13 Mean learning process ($\mathcal{E}$ = 0.8, map2)

DOI:10.5963/IJCSAI0303004

TABLE 5 MEAN (AND MEDIAN) OF THE NUMBER OF LEARNED ACTIONS DURING 500 EPISODES (100 AGENTS, MAP2)

ε	QL	SRL	SRL64	CFRL64 (Shift = 1)	CFRL64 (Shift = 2)
0.05	128.82 (44)	117.96 (44)	90.04 (44)	93.16 (44)	80.40 (44)
0.1	146.80 (44)	119.28 (44)	111.44 (44)	73.04 (44)	80.00 (44)
0.2	123.44 (44)	84.22 (44)	94.96 (44)	73.02 (44)	72.20 (44)
0.4	98.32 (44)	87.06 (44)	218.18 (150)	64.70 (44)	70.20 (44)
0.8	44.00 (44)	44.78 (44)	755.30 (853)	51.34 (44)	53.82 (44)

C. Discussion

The results implied that SRL works as well as normal Q-learning in goal search (Perfectly Observable Markov Decision Processes). However, some SRL agents do not work well. In contrast, our mechanism cannot be affected by action selection policy more than SRL (Tables 4 and 5; Figures 13-15, 18-22).

Our method (cyan line) brought better results than the existing method (red line) for 64 states of memory agents. In particular, if an agent uses exploratory action selection policy (like ε =0.4, 0.8), the existing method lost good state-action sets instead of undetermined state-action sets. On the other hand, CFRL yielded almost as good results as Q-learning in any case in spite of its small memory.

If an agent does not have enough memory to learn, it gets lost on the way at the first part of each episode. This is because SRL stocks *Value*(*s*, *a*) in goal-nearest order preferentially.

To investigate the sensitivity of the parameter “Shift in CFRL”, we changed the parameter and experienced. However, it made few differences. It should be important not to destroy the best way in SRL, thus, shift parameter has less influence.

Our attempt resulted in a great improvement in the reinforcement learning for agents without large memory.

VI. CONCLUSIONS

We improved a novel reinforcement learning method (SRL) by low-priority cut-in (moratorium). SRL categorizes several state-action sets as “GOOD" and arranges alignment sequences of estimated values as their priorities, but is largely influenced by the agent's policy.

Our newly proposed method is applied to several simulations (Goal search (Perfectly Observable Markov Decision Processes)) in order to demonstrate the adaptability of the model. Experimental results showed that our method is as useful as normal Q-learning under any circumstance and placing a moratorium is effective.

Comparative study with other learning approaches remains for future work.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Education, Culture, Science and Technology, Japan under Grant-in-Aid for Scientific Research No. 21700242.

REFERENCES

[1] C. Szepesvari: *Algorithms for Reinforcement Learning*, Morgan and Claypool, 2010.

[2] T. Jaakkola, M. Jorban, and S. Singh: "On the convergence of stochastic iterative dynamic programming algorithms", Neural Computation, vol. 6, iss. 6, pp. 1185-1201, 1994.

[3] C. J. C. H. Watkins and P. Dayan: ``Technical Note: Q-learning", Machine Learning, vol. 8, pp. 279-292, 1992.

[4] S. Katayama and S. Kobayashi: ``A Logarithmic-Time Updating Algorithm for TD(λ) Learning", Journal of Japanese Society for Artificial Intelligence, vol. 14, iss. 5, pp. 879-890, 1999.

[5] A. Notsu, K. Honda, H. Ichihashi and Y. Komori: ``Simple Reinforcement Learning for Small-Memory Agent", 10th International Conference on Machine Learning and Applications, vol. 1, pp. 458-461, 2011.

[6] A. Notsu, K. Honda and H. Ichihashi: ``Proposal for Notion Learning of Reinforcement Learning", Social Intelligence Design 2009, T1-3, 2009.

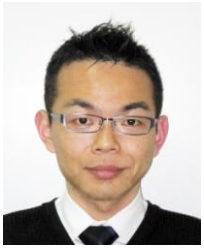
[7] A. Notsu, K. Honda and H. Ichihashi: ``Particle Swarm for Reinforcement Learning", Joint 5th International Conference on Soft Computing and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems, pp. 809-812, 2010.

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

[9] L. Sttel and P. Vogt: ``Grounding Adaptive Language Games in Robotic Agents", The Fourth European Conference on Artificial Life (ECAL '97), pp. 474-482, 1997.

[10] M. Tan: ``Multi-agent Reinforcement Learning: Independent vs. Cooperative Agent", The 10th International Conference on Machine Learning (ICNL' 93), pp. 330-337, 1993.

DOI:10.5963/IJCSAI0303004



Akira Notsu received the B.E., M.I. and D. Informatics degrees from Kyoto University in 2000, 2002, and 2005, respectively.

He is currently an Associate Professor, Department of Computer Science and Intelligent Systems, Osaka Prefecture University. His research interests include agent-based social simulation, communication networks, game theory, human–machine interface, and cognitive engineering.

Yuki Tezuka has been with the Graduate School of Engineering, Osaka Prefecture University. His research interests include reinforcement learning and multi-agent simulation.

Katsuhiro Honda received the B.E., M.E., and D.Eng. degrees in industrial engineering from Osaka Prefecture University, Osaka, Japan in 1997, 1999, and 2004, respectively.

From 1999 to 2013, he was a Research Associate, Assistant Professor and Associate Professor at Osaka Prefecture University, where he is currently a Professor in the Department of Computer Sciences and Intelligent Systems. His research interests include hybrid techniques of fuzzy clustering and multivariate analysis, data mining with fuzzy data analysis, and neural networks.

DOI:10.5963/IJCSAI0303004