An Improved Domain-Knowledge-based Reinforcement Learning Algorithm

Si Young Jang* and Il Hong Suh*

* Intelligent Control and Robotics Lab., School of Electrical Engineering and Computer Science,
Hanyang University, 1271 Sa-1 Dong, Sangrok-Gu, Ansan-Si, Kyungki-Do, 426-791, South Korea
(Tel : +81-31-408-5802; E-mail: ihsuh@hanyang.ac.kr)

Abstract: If an agent has a learning ability using previous knowledge, then it is expected that the agent can speed up learning by interacting with environment. In this paper, we present an improved reinforcement learning algorithm using domain knowledge which can be represented by problem-independent features and their classifiers. Here, neural networks are employed as knowledge classifiers. To show the validity of our proposed algorithm, computer simulations are illustrated, where navigation problem of a mobile robot and a micro aerial vehicle(MAV) are considered.

Keywords: reinforcement learning, domain knowledge, problem-independent feature, neural network, classifier

1. INTRODUCTION

Reinforcement learning shown in Figure 1.1 addresses how to choose the optimal sequence of actions to achieve goals through iterative processes that agent performs an action in its environment and gets a reward for action[4].

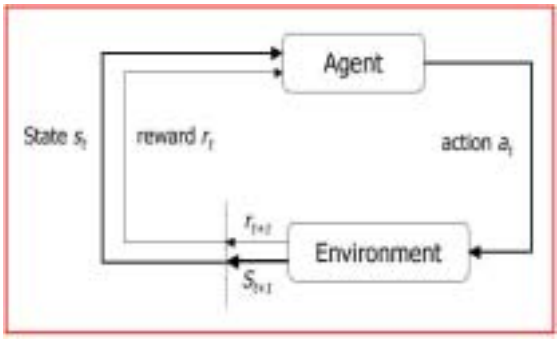

Figure 1.1 Reinforcement Learning

Recently, several types of reinforcement learning algorithms have been applied to path planning of a mobile robot[8], biomimetics and artificial life[11], analysis of game theory[3], packet routing of networks[12], and control of robot manipulators[7].

Q-learning is one of the most widely used reinforcement learning algorithms[5,6]. It can choose an optimal action in unknown environment, but has a few defects. First, Q-learning algorithm requires many iterations to achieve a goal and needs large memories, because the state and action space has been implemented by sampling continuous space as a lot of discrete states. Several alternative methods have been proposed to address this problem where real number reward method[14,16], Prioritized Sweeping[15], and region-based Q-learning could be included.

Second problem is "scratch" which could be defined as initialization of the Q-table in the beginning of learning. Because optimality of a sequence of actions obtained by the Q-learning holds only on a given specific problem, for another given problem, agent has to learn optimal sequence of actions from scratch. The advantage of reinforcement learning is to allow the agent to learn approximately optimal solutions of a given problem through stochastically similar sweeping iterations. Unfortunately, on the other hand, because of this stochastic optimal search from scratch, even if the agent, which learned optimal solution of a problem previously, has been given the same problem, it should take almost same number of iterations to learn the Q-table.

If an agent has a prior knowledge from previously solved problems, the agent can speed up learning on new similar problems in the sense of reduction of a large number of training episodes. Many researchers proposed reinforcement learning methods using a priori knowledge such as adaptive Q-table method changing the size of Q-table[14], policy reuse method using some of previous action functions[13], and dyna-Q method using a virtual model of agent environment[4]. The studies on integrating advice of agent into neural networks for an effective learning, appending advices of external observer[1,2,9] or assessing the expertness values of other agent knowledge[10] have been suggested as well.

Most of those methods could speed up learning by using the knowledge about environment obtained from previous experiences. The reinforcement learning comes simply from the Pavlovian conditioning of animal learning, while the method to use the knowledge about environment comes from the advanced learning; storing something in memories, and solving new problem relatively easy based on the memorized knowledge[17].

As research works on reinforcement learning using knowledge about environment, there are methods of Maclin and Shavlik[9], Singer and Veloso[3], Ahmadabadi and Asadpour[10]. Among them, Maclin and Shavlik proposed a method which converted advices into statements such as if-then conditions, which were stored in a neural network named as KBANNs (Knowledge BAsed Neural Networks). Here, the advices are made from optimal actions for a domain which can be defined by sensing radius. This method enables the agent to take advantage of external advices, and thus to learn some sequences of action to achieve sub-goals. Singer and Veloso[3] have made advices by using Local State Features to express local states of a given environment, and have proposed a method for agent to integrate advices into its own knowledge by appending nodes into trees or by training back-propagation neural networks. Ahmadabadi and Asadpour[10] have studied how agent can choose proper actions from not only its own experience but also other agent experiences. To do this, they have addressed how to evaluate the expertness value of other agent experience. And then, they have suggested a way for agent to choose actions from linear combination of these expertness-based weighted values But, the above proposed methods may show local minima since the knowledge depends on local sensing radius of an agent.

In this paper, we present an improved reinforcement learning algorithm using domain knowledge. In our algorithm, (1)The variable utilization rate of classifier for efficient exploration is to be designed in such a way that the rate value

is given as high at the beginning phase, and the rate value exponentially decreases as the episode increases during the transient phase, and finally it is given as low at the convergent phase. (2)Number of Examples is to be reduced by modifying the mechanism to append new Examples to an Example Set. (3)Direction-dependent multiple classifiers are to be designed to store more problem-independent information. To show the validity of our proposed method, several computer simulation results are illustrated, where navigation problems of a mobile robot and a micro aerial vehicle(MAV) are considered.

2. DIRECT APPLICATOIN OF REINFORCEMENT LEARNING USING GENERALIZED DOMAIN KNOWLEDGE TO NAVIGATION PROBLEM

Figure 2.1 shows a conceptual diagram of the reinforcement learning using domain knowledge proposed by Singer and Veloso[3]. In their work, they applied domain knowledge to the game such as SOKOBAN whereas in our work, we will apply domain knowledge to a navigation problem. To do this, in this section, we redefine "Local State Feature" and classifier to be applied to a navigation problem. Specifically, domain knowledge is information about environment with which an agent interacts. And it is expressed as a set of tuples {"Local State Feature", action, evaluation}. A classifier is designed or trained to contribute to effective exploration by employing such domain knowledge. In this paper, a classifier is implemented as neural networks.

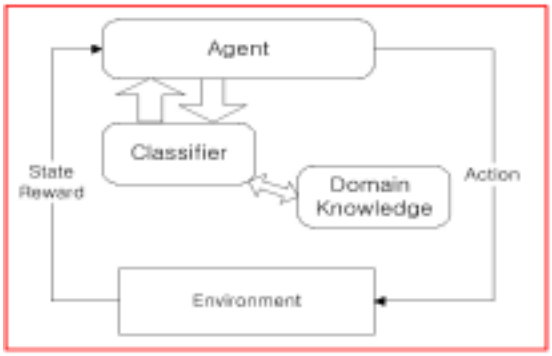

Figure 2.1 Reinforcement Learning using Domain Knowledge

States used in classical reinforcement learning algorithms are often defined based on physical location of an agent. But here, "Local State Feature" is made to be independent of location in order to reduce problem-by-problem dependence of agent knowledge.

2.1 Classifier design for generalization of domain knowledge

In this section, a method how an agent could learn domain knowledge is described. To begin with, given a problem, the agent optimally updates Q-table from a problem. This Q-table is a set of action value functions corresponding to each state in the environment with which the agent interacts. And then, an Example which is a basis of domain knowledge is extracted from the learned Q-table and appended to Example Set. Here, an Example is defined as a structure that consists of a tuple {"Local State Feature", action for each state, the evaluation for the action}.

"Local State Feature" of each state is utilized as action firing conditions for current state and its neighbor states of the agent. As shown in Figure 2.2, "Local State Feature" is composed of sensing information whether state condition is a wall or a floor. A set of Example is shown in Figure 2.3. There are two types of Examples from the learned Q-table according to a criterion: One is the Positive Example that is evaluated as a positive experience in the sense that action for the current state let the agent move toward the goal. Later if an agent meets a state corresponding to the Positive Example, the agent would be indicated to do the action given by the Example. The other is the Negative Example which is evaluated as a negative experience in the sense that current action for the present state let the agent move away from the goal. Later if an agent meets a state corresponding to the Negative Example, the agent would be indicated not to do the action by the Example.

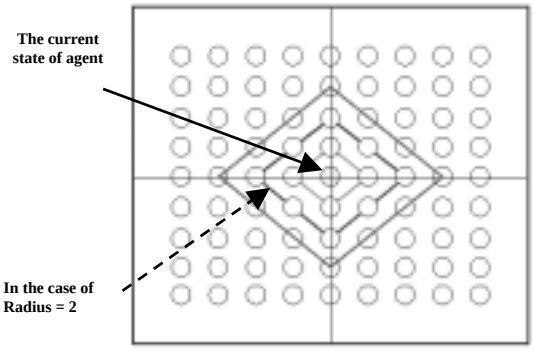

Figure 2.2 Exemplar Sensing radius of an Agent

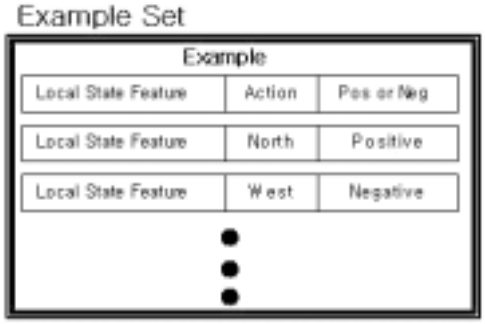

Figure 2.3 Composition of Example Set

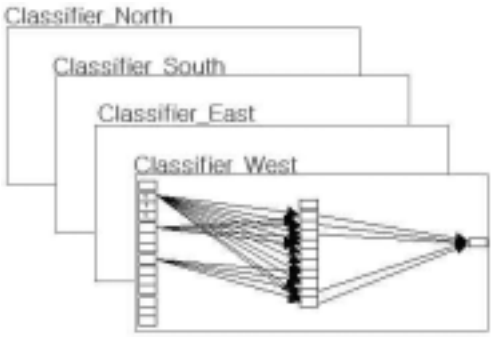

Figure 2.4 Structure of Classifiers

Figure 2.4 shows the structure of classifiers. Each classifier is expressed as a neural network, and is separately defined

according to each action. To learn domain knowledge including the Example Set, the agent chooses classifier corresponding to an action defined within Example Set. Then the feature value of "Local State Feature" is applied to the network as an input, and a large value(0.9) in assigned as the output value in the case of Positive Example. Or a small value(0.1) is assigned as output value of the network in the case of Negative Example.

2.2 Use of Domain Knowledge for Exploration

The agent could use generalized domain knowledge which would be stored in the learned classifier in the phase of exploration to effectively solve a given new problem.

Generally, the action policy of standard Q-Learner is given as

$$\pi^*(s) = \arg\max_a Q(s,a). \tag{2.1}$$

Let a normalized action value function be denoted by

$$P(a,s) = \frac{Q(s,a)}{\sum_{a'} Q(s,a')}. \tag{2.2}$$

And, here $C_a(s)$ is defined as the output of classifier C_a for a state(s). Then a normalized output of classifier C_a is written as

$$w(s,a) = \frac{C_a(s)}{\sum_{a'} C_{a'}(s)}. \tag{2.3}$$

Eq.(2.2) and Eq.(2.3) are combined by the Boltzmann exploration method for the agent to use the classifier advices in exploration. The combined policy is expressed by

$$\pi^*(s) = \arg\max_a \frac{w(s,a)\cdot P(s,a)}{\sum_{a'} w(s,a')\cdot P(s,a')}. \tag{2.4}$$

The agent by using the combined policy in Eq.(2.4) is expected to choose proper actions to achieve a goal in phase of exploration in the sense that the agent selects an action between an action of reinforcement learner and an advice of classifier.

2.3 Some considerations of reinforcement learning using domain knowledge

Followings are something to be considered for efficient learning of classifier and for improvement of learning speed of reinforcement learner using domain knowledge.

1. The important contribution of domain knowledge is to give an agent advices to achieve a goal in the beginning phase of exploration just after scratching. And, as the Q-table of the agent is to be optimal, on the contrary, domain knowledge may prevent the agent from learning optimal actions. Thus, the variable utilization rate of domain knowledge will be needed.
2. Learning speed of a neural network depends on the number of Examples within Example Set. If the size of Example Set can be reduced, the learning of the agent will be faster than before.
3. In the work of Singer and Veloso[3], "Local State Feature" was defined to be applied to games such as SOKOBAN whereas it is not proper for navigation or path-planning problem without information of moving direction. We found ambiguous situations that the agent can not make out whether current action of the agent is correct to achieve a goal or not. Moreover, these ambiguous situations incurred larger number of training steps than classical reinforcement learning. An instance of these situations is shown in Figure 2.5. The action to be recommended by the classifier under the learning experience of Figure 2.5(a) should be the action toward the east or the north direction.

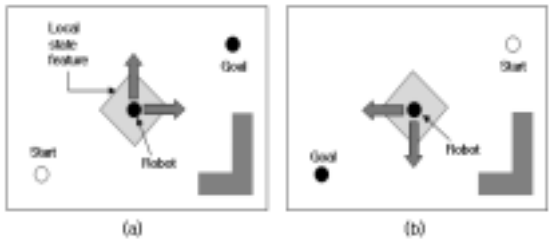

Figure 2.5 Example of ambiguous problems

Suppose that the agent is in the situation of Figure 2.5(b). Then, the recommended action(east or north) of classifier would prevent the agent from achieving a goal. Thus, the improved classifier system to learn domain knowledge including moving direction will be needed.

3. AN IMPROVED REINFORCEMENT LEARNING ALGORITHM USING DOMAIN KNOWLEDGE

3.1 Variable rate of utilization of classifier

The advantage of the method using domain knowledge is to advise the agent to do proper actions at the beginning phase. But this method may prevent the agent from learning optimal actions at convergent phase. Thus, we modify the utilization rate of domain knowledge to improve this problem. The action policy equation given by Eq.(2.4) is modified as

$$\pi^*(s) = \arg\max_a \frac{w_{new}(s,a)\cdot P(s,a)}{\sum_{a'} w_{new}(s,a')\cdot P(s,a')} \tag{3.1}$$

$$where \quad w_{new}(s,a) \leftarrow w(s,a) + Value_{weight}(episode).$$

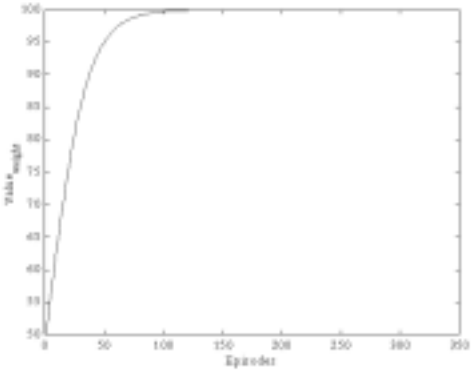

Figure 3.1 Graph of $Value_{weight}(episode)$

In Eq.(3.1), $w_{new}(s,a)$ is the normalized output value of classifiers , and $Value_{weight}(episode)$ is an exponential-type function shown in Figure 3.1. As episode increases, modified

action policy Eq.(3.1) has become Watkins's action policy equations. Thus the convergence of this method is shown as

$$w_{new}(s,a) \approx Value_{weight}(episode), \qquad \pi^*(s) \approx \arg\max_a \frac{P(s,a)}{\sum_{a'} P(s,a')},\ as\ \ episode \to \infty. \tag{3.2}$$

3.2 Example-appending mechanism

Reinforcement learning algorithm using domain knowledge by Singer and Veloso[3] generates Positive and Negative Examples for every state, and appends all of them to Example Set to be used to train neural networks. But, actually useful Examples for domain knowledge come from the states around path from the starting state to the goal state. In this sense, we make Example Set not by using every state in the state space, but by employing examples only the states around path from the starting state to the goal state.

3.3 Direction-dependent multiple classifier

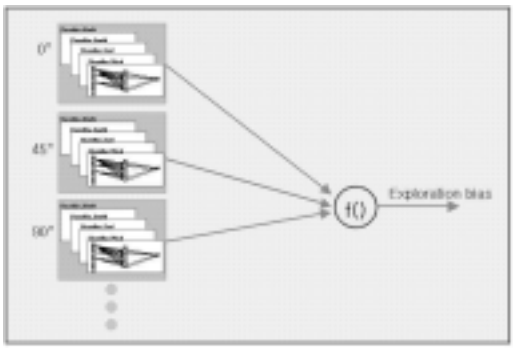

Figure 3.2 Improved classifier structure

As we mentioned in section 2.3, there are many ambiguous situations while applying reinforcement learning using domain knowledge, if "Local State Feature" is dependent on moving direction of an agent. Thus, we suggest the classifier structure with direction-by-direction classifiers. Here, we design eight classifiers for eight directions. The agent could know the goal direction though the first iteration. According to this goal direction, the agent is to decide which classifier will be used. Figure 3.2 shows the conceptual structure of the proposed algorithm.

Let $P(s,a)$ be the output of classifier and let $P'(s,a)$ be linear combination of each classifier written as

$$P'(s,a_i) = \sum_j P_i(s,a_i) \cdot W_j \ , \tag{3.3}$$

where a_i implies the *i* th action, and $W_j(x)$ is given as

$$W_j(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-j}{\sigma}\right)^2\right), \ x: direction\ angle \tag{3.4}$$

Finally, the action policy of our improved reinforcement learning algorithm using domain knowledge is expressed as

$$\pi^*(s) = \arg\max_a \frac{w_{new}(s,a) \cdot P'(s,a)}{\sum_{a'} w_{new}(s,a') \cdot P'(s,a')} \tag{3.4}$$

where $P'(s,a) = \sum_i P_i(s,a) \cdot W_i$.

4. SIMULATION

4.1 Simulation Environment

4.1.1 Mobile Robot

For simulation, the type of neural network is chosen as one hidden layer network, and the number of units to each layer is calculated to 121, 20, and 1. For mobile robot navigation, 224 examples such as Figure 4.1 are chosen to train classifiers.

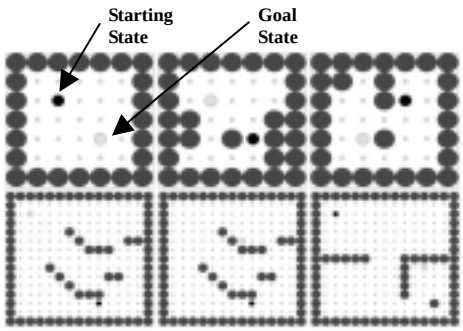

Figure 4.1 Examples of many kinds of maps

4.1.2 Micro Aerial Vehicle(MAV)

MAV example is shown as in Figure 4.2. The number of units to each layer of neural network is calculated to 257, 20, and 1. 1560 examples are chosen to train classifiers.

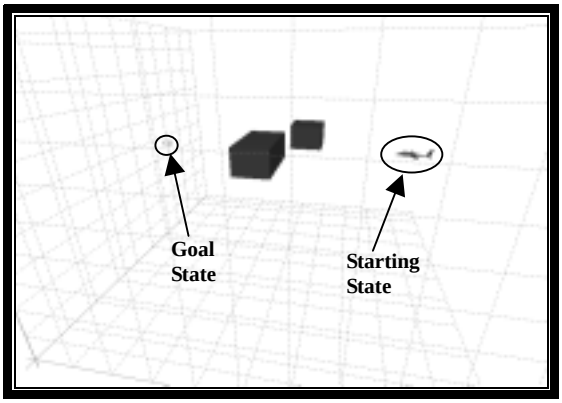

Figure 4.2 Example of Map for MAV

Simulation environment of MAV is three- dimensional space. Thus, direct application of Gaussian distribution to sphere is impossible due to curse of dimensionality. Here, we used linearly approximated distance between direction vector and center vectors of Gaussian distribution as shown in Figure 4.3.

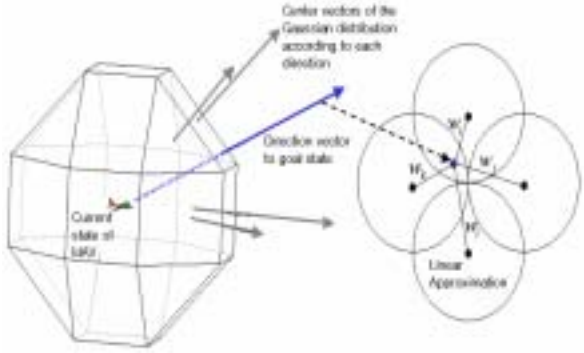

Figure 4.3 Approximated Gaussian weight value for MAV

4.2 Simulation Result

Figure 4.4 shows the simulation result to compare reinforcement learning using domain knowledge with reinforcement learning without using domain knowledge. In bold ellipse of Figure 4.4, we can observe that remarkable reduction of path length could be obtained at the beginning phase when domain knowledge is applied.

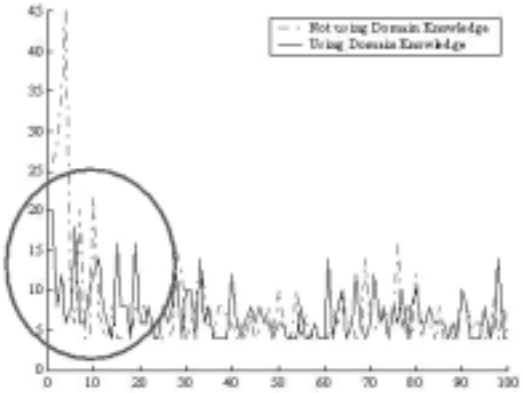

Figure 4.4 Comparison of using Domain Knowledge and not using Domain Knowledge

But reinforcement learning using domain knowledge may be not effective in coping with ambiguous problems shown as Figure 2.5. Due to the incorrect advice of classifiers, the path length would become long as shown in Figure 4.5.

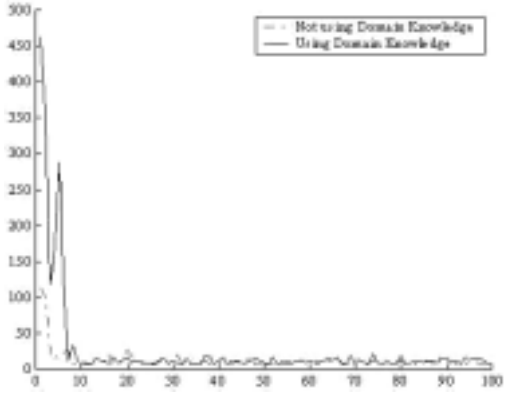

Figure 4.5 Result due to ambiguous problem

Figure 4.6 shows the simulation results of reinforcement learning using our improved domain knowledge. Since direction-dependent classifiers advised the agent to select proper actions to achieve the goal, performance has been enhanced.

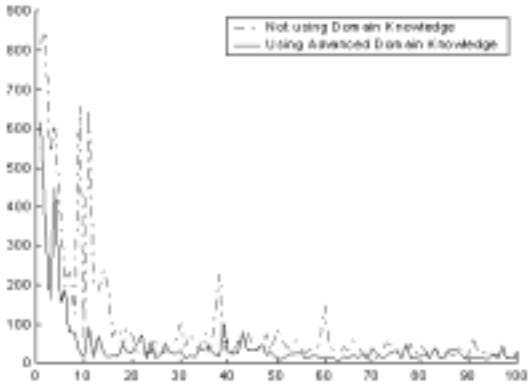

Figure 4.6 Comparison of using Domain Knowledge and not using Improved Domain Knowledge

Figure 4.7 and Figure 4.8 show the averaging performance of our improved domain-knowledge-based reinforcement learning algorithm, domain-knowledge-based reinforcement learning algorithm, and classical reinforcement learning algorithm. Those are applied to the navigation problems of the mobile robot and the MAV. Many exemplar problems were selected to have generality where the goal direction and start position are randomly distributed. In case of mobile robot, shown in Figure 4.7, the reinforcement learning using domain knowledge by Singer and Veloso shows performances not better than the classical reinforcement learning, whereas our proposed algorithm shows similar performance to the classical reinforcement learning.

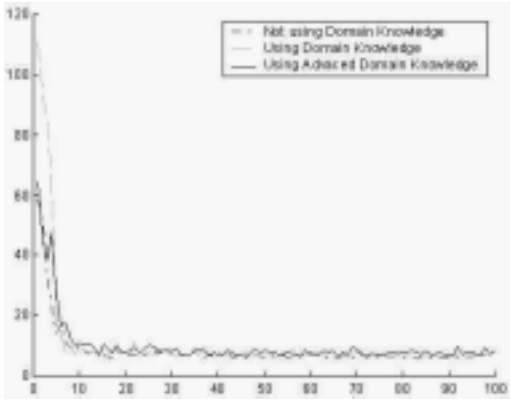

Figure 4.7 Comparison of averaging performance applied to mobile robot

With domain knowledge including ambiguous features, reinforcement algorithm is not applicable to large state and action space such as MAV three-dimensional space. Figure 4.8 shows the averaging performance of reinforcement learning using our improved domain knowledge is superior to that of classical reinforcement learning, when those algorithms are applied to MAV path planning problems.

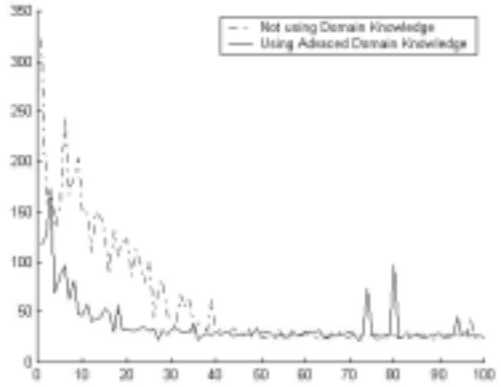

Figure 4.8 Comparison of averaging performance applied to MAV

5. CONCLUSIONS

In this paper, we proposed an improved reinforcement learning algorithm using domain knowledge which could be obtained from previously solved problems. Our suggested algorithm shows more robust performance and faster convergence speed than the algorithm in [3]. On the other hand, our proposed method requires classifiers as many as the number of directions and therefore it may take more times to train neural networks classifiers.

REFERENCES

[1] S. Y. Jang, I. H. Suh, et al., "Reinforcement Learning Algorithm Using Domain Knowledge," *Proc. of the international Conference on Control Automation and Systems*, October, 2001.

[2] B. O. Kim, I. H. Suh, et al., "Reinforcement Learning Algorithm using Domain Knowledge for MAV," *Proc. of the KIEE Summer Annual Conference*, July, 2002.

[3] B. Singer, M. Veloso, "Learning State Feature from Policies to Bias Exploration in Reinforcement Learning," *Technical Note of Carnegie Mellon University*, April, 1999.

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

[5] C. Watkins, P. Dayan, "Q-Learning, technical note," *Machine Learning*, Vol. 8, pp. 279-292, 1992.

[6] C. Watkins, "Learning from Delayed Reward," *PhD Thesis, Cambridge*, May, 1989.

[7] I. H. Suh, J. H. Kim and S. –R. Oh, "Region-based Q-Learning for Intelligent Robot Systems," *Proc. of IEEE Int. Conf. on Computational Intelligence in Robotics and Automation*, pp. 163-178, July, 1997.

[8] J. H. Cha, S. H. Kong, I. H. Suh, "Region-based Q-learning For Autonomous Mobile Robot Navigation," *KACC2000, 15th Korea*, October, 2000.

[9] R. Maclin, J. W. Shavlik, "Creating Advice-Taking Reinforcement Learners," *Machine Learning*, 22, 1996.

[10] M. Nili Ahmadabadi, M. Asadpour, "Expertness Based Cooperative Q-Learning," *IEEE Trans. On Systems, Man, And Cybernetics-Part B: Cybernetics*, Vol 32, No. 1, February. 2002.

[11] S. Yoon, "Affective Synthetic Characters," *PhD Thesis, MIT*, June, 2000.

[12] M. Littman and J. Boyan, "A distributed reinforcement learning scheme for network routing," *Technical Report, School of Computer Science, Carnegie Mellon University*, 1993.

[13] M. Bowling and M. Veloso, "Bounding the suboptimality of reusing subproblems," *In Proceeding of the NIPS Workshop on Abstraction in Reinforcement Learning*, December, 1998.

[14] Y. Hirashima, "Q-Learning Algorithm Using an Adaptive-Sized Q-table," *Proceeding of the 38th Conf. on Decision & Control, Phoenix*, pp.1599-1604, December, 1998.

[15] A. W. Moore, C. G. Atkeson, "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time," *Machine Learning*, 13, 1993.

[16] T. D'Orazio, G. Cicirelli, "Continuous Reward versus Discrete Reward in a Q-learning Agent," *The Fifth ICARCV'98, Singapore*, December, 1998.

[17] R. S. Sutton and A. G. Barto, "Time-derivative models of Pavlovian Reinforcement," *Learning and Computational Neuroscience: Foundations of Adaptive Networks, pp. 497-537, Cambridge, MA: MIT Press*, 1990.