We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

9

Hierarchical Reinforcement Learning Using a Modular Fuzzy Model for Multi-Agent Problem

Toshihiko Watanabe
Osaka Electro-Communication University
Japan

1. Introduction

Reinforcement learning (Sutton & Barto, 1998; Watkins & Dayan, 1998; Grefenstette, 1988; Miyazaki et al., 1999; Miyazaki et al., 1999) among machine learning techniques is an indispensable approach to realize the intelligent agent such as autonomous mobile robots. The importance of the technique is discussed in several literatures. However there exist a lot of problems compared with the other learning techniques such as Neural Networks in order to apply reinforcement learning to actual applications. One of the main problems of reinforcement learning application of actual sized problem is "curse of dimensionality" problem in partition of multi-inputs sensory states. High dimension of input leads to huge number of rules in the reinforcement learning application. It should be avoided maintaining computational efficiency for actual applications. Multi-agent problem such as the pursuit problem (Benda et al., 1985; Ito & Kanabuchi, 2001) is typical difficult problem for reinforcement learning computation in terms of huge dimensionality. As the other related problem, learning of complex task is not easy essentially because the reinforcement learning is based only upon rewards derived from the environment.

In order to deal with these problems, several effective approaches are studied. For relaxation of task complexity, several types of hierarchical reinforcement learning have been proposed to apply actual applications (Takahashi & Asada, 1999; Morimoto & Doya, 2000). To avoid the curse of dimensionality, there exists modular hierarchical learning (Ono & Fukumoto, 1996; Fujita & Matsuno, 2005) that construct the learning model as the combination of subspaces. Adaptive segmentation (Murano & Kitamura, 1997; Hamagami et al.,2003) for constructing the learning model validly corresponding to the environment is also studied. However more effective technique of different approach is also necessary in order to apply reinforcement learning to actual sized problems.

In this chapter, I focus on the well-known pursuit problem and propose a hierarchical modular reinforcement learning that Profit Sharing learning algorithm is combined with Q Learning reinforcement learning algorithm hierarchically in multi-agent environment. As the model structure for such huge problem, I propose a modular fuzzy model extending SIRMs architecture (Seki et al., 2006; Yubazaki et al., 1997). Through numerical experiments, I show the effectiveness of the proposed algorithm compared with the conventional algorithms.

The chapter is organized as follows. In section 2, an overview of pursuit problem as multi-agent environment is presented. In section 3, I propose construction of agent model and essential learning algorithms of a hierarchical reinforcement learning using a modular model architecture. In section 4, I propose a modular fuzzy model for agent model construction. The results of numerical experiments are shown in section 5. Finally, conclusions are drawn in section 6.

2. Pursuit problem as multi-agent environment

The pursuit problem is well known and has been studied as typical benchmark problem in Distributed Artificial Intelligence research field (Benda et al., 1985). It is multi-agent based problem that hunter agents act collaboratively to capture prey agent. Figure 1 shows the 4-agent pursuit problem in 7×7 grids field. In the problem, all agent behave in turn to move upward, downward, rightward, leftward in one gird, or to stay. Collision of the agents is prohibited because one grid allows only one agent to stay. The objective of the simulation is to surround the prey agent by the hunter agents as shown in Fig.2.

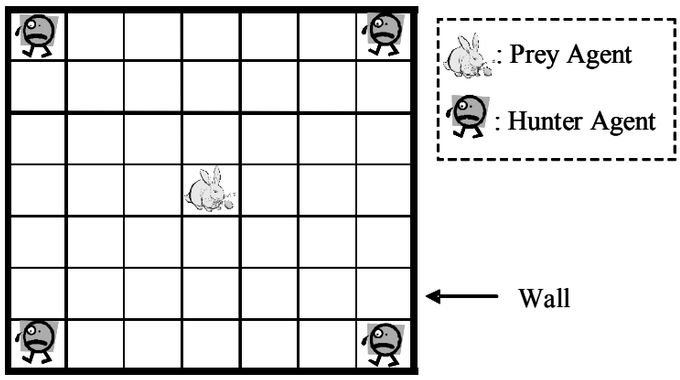


Fig. 1. 4-Pursuit Problem(7x7 grids)

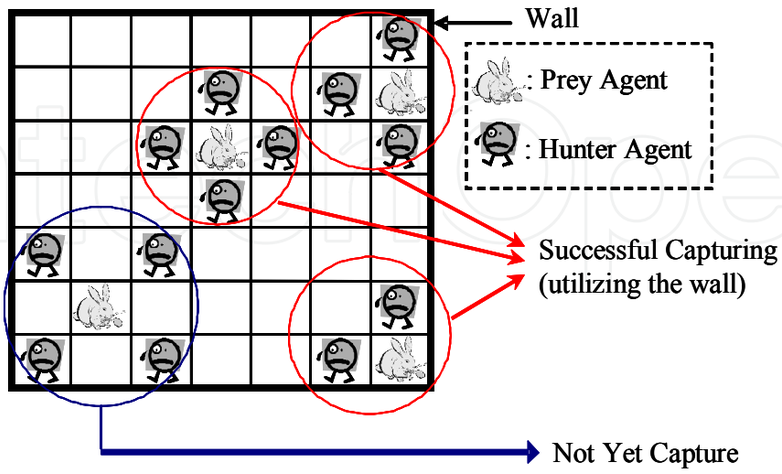


Fig. 2. Examples of Capturing Condition in Pursuit Problem

The hunter agents can utilize walls for surrounding as well as surrounding by whole hunter agents. When the surrounding is successfully performed, related hunter agents receive

reward from the environment to carry out reinforcement learning. As for behavior of the prey agent, it behaves to run away from the nearest hunter agent for playing a fugitive role. For actual computer simulations or mobile robot applications, it is indispensable to avoid huge memory consumption for the state space, i.e. "curse of dimensionality", and to improve slow learning speed caused by its sparsity(e.g. acquired Q-value through reinforcement learning). In this study, I focus on the 4-agent pursuit problem to improve precision and efficiency of reinforcement learning in multi-agent environment and to demonstrate settlement of "curse of dimensionality".
For simulation study, I adopt "soft-max" strategy for selecting the action of the hunter agents. The conditional probability based on Boltzmman distribution for action selection is as follows:

$$p(a \mid s) = \frac{\exp\left(w(s,a)/T_t\right)}{\sum_{d \in N} \exp\left(w(s,d)/T_t\right)} \quad , \; T_{t+1} = T_t \times \beta \tag{1}$$

where T_t is temperature at t-th iteration, s is state vector, a is the action of the agent, β is the parameter for temperature cooling($0<\beta<1$), w denotes evaluation value for state-and-action pair, and N denotes the set of all alternative action at the state s. Owing to this mechanism, the hunter agent act like random walk(exploring) with high temperature value in the early simulation trials and act definitely based on acquired evaluation values in the later simulation trials according to the lowered temperature value.

3. A hierarchical reinforcement learning using modular model architecture

3.1 Basic concepts

There exist two problems to solve the pursuit problem efficiently. One is huge memory consumption for internal knowledge expression of the agents expressed as evaluation weights corresponding to the pair of state-and-action caused by the grid size of the environment and the number of hunter agents. In order to restrain the increase of required memory for the agents, modular structure is applied for expression of the agent knowledge base. The other is complex objective, i.e. surrounding the prey *collaboratively*. In general, it is effective for dealing with such complex task to decompose into sub-tasks. Then I decompose the task into hierarchical sub-tasks to fulfill reinforcement learning effectively. I propose a hierarchical modular reinforcement learning to solve the above described two problems in the multi-agent pursuit simulation.

3.2 Hierarchical task decomposition for agent learning

It is difficult to decide how many kinds of subtask should be decomposed into. In this study, I empirically decompose the surrounding task(capturing) into "decision of move position target" for surrounding according to current monitored state and "selection of appropriate action" to move to the target position of each agent. The latter task is native, isolated from the other hunter agents, and is not needed to be collaborative such as position control of the single agent. In other words, the task is decomposed into "surrounding" task synchronized with the other hunter agents and "exploring the environment" task. Moreover, the upper task corresponds only to collaborative surrounding strategy. Figure 3 shows the internal hierarchical structure of the hunter agent. The knowledge base of the agent is composed of the "Rules in Upper Layer" and the "Rules in Lower Layer" as shown in the figure. It is

important to keep learning capability as well as task decomposition. According to the two-layered decomposition, rules in the lower layer can be adapted corresponding to the agent behavior in every step as Markov Decision Process, as shown in Fig.4.

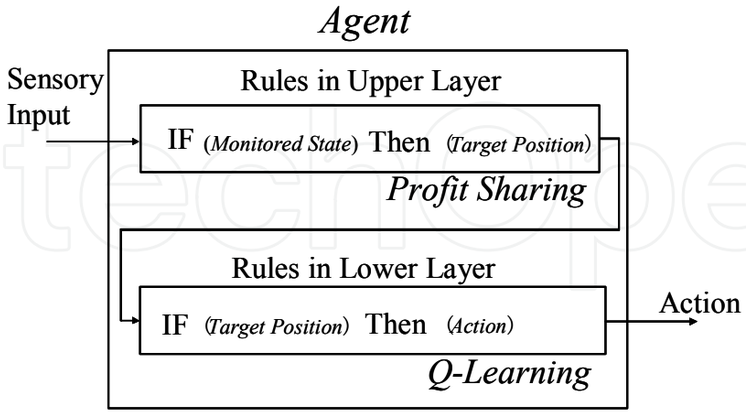


Fig. 3. Internal Hierarchical Structure of Hunter Agent

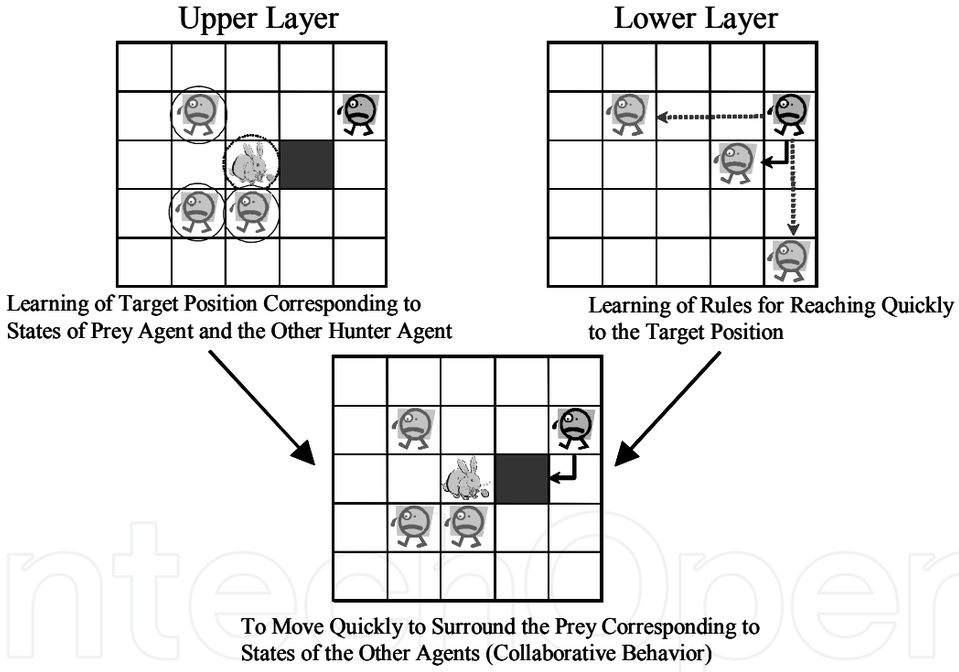


Fig. 4. Conceptual Diagram of Hierarchical Task Decomposition

3.3 A modular profit sharing learning for upper layer

In the upper layer, the target position of the agent is decided based on observed state such as the current position of the prey agent and the other hunter agents. The rules in the upper layer express goodness of the target position corresponding to the current state excluding actual actions. In order to construct the rules based on the current state combination, huge corresponding memory is needed. To avoid such requirement, the authors applied modular structure for the rule expression (Takahashi & Watanabe, 2006) in the upper layer as shown in Fig.5. In this section, the dimension of modular model is assumed to be three for

explanation simplicity. Higher dimension can also be considered as the same manner. Original state space of each agent is expressed as the modular model by covering with three subspaces of oneself-and-another pair as shown in Fig.6.

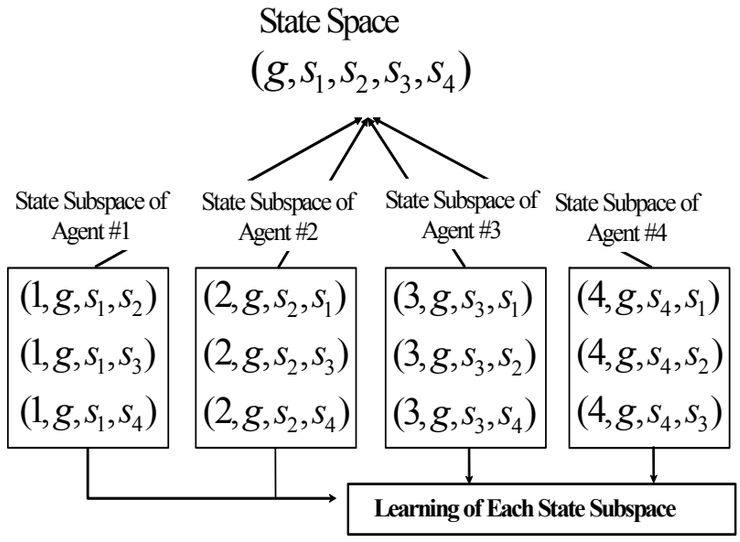


Fig. 5. Modular Structure of Agent State Maps

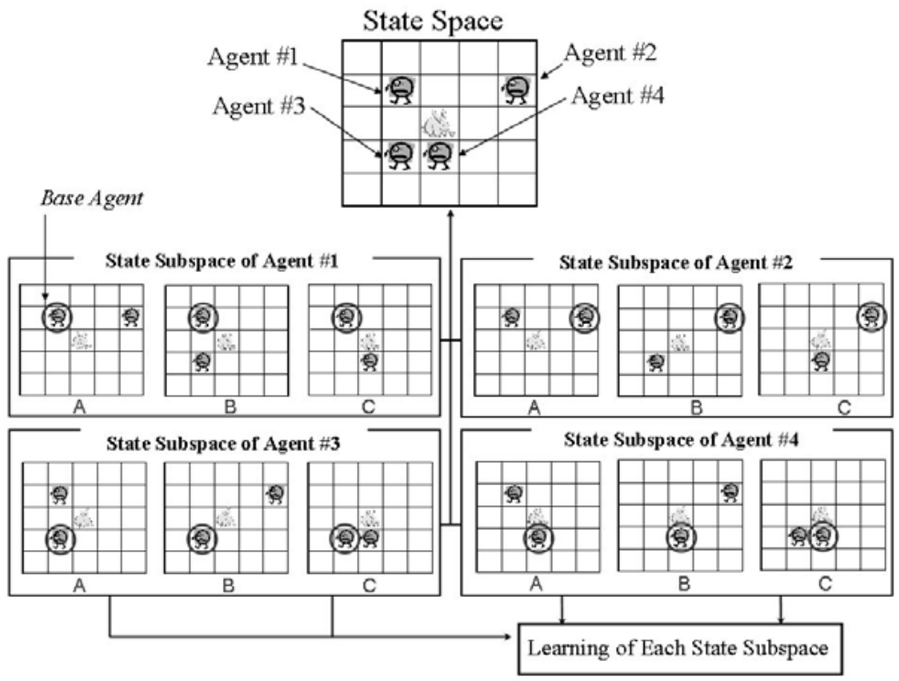


Fig. 6. An Example of Modular Structured Maps

The weights of rules in the upper layer are updated by Profit Sharing learning algorithm(Miyazaki et al., 1999), when capturing succeeds, as the following formulations:

$$
\begin{aligned}
u(e,g_i,h_{e,i},h_{\varepsilon,i}) &= u(e,g_i,h_{e,i},h_{\varepsilon,i}) + k(e,g_i,h_{e,i},h_{\varepsilon,i}) \\
k(e,g_{i+1},h_{e,i+1},h_{\varepsilon,i+1}) &= \frac{1}{\rho}k(e,g_i,h_{e,i},h_{\varepsilon,i}) \qquad (i=0,1,\ldots,m-1,\ \varepsilon \neq e)
\end{aligned}
\tag{2}
$$

where u is the weight of the rule, g is state of the prey agent, $h_{e,i}$ denotes the state of agent e at i step ago from the current step, k denotes the reinforcement function, and ρ is the parameter.

In the action phase, the target position is desirable to be decided as a sub-goal for surrounding task instead of final goal corresponding to the current state of the prey agent according to the rule weights. In this study, the target position of the agent is generated as:

$$
p = \arg\max_{v} \sum_{q} \frac{u(e,g,v,h_q)}{\mu^{\|h_e - v\|}} \qquad (q \neq e, \mu \geq 1) \tag{3}
$$

where h_e denotes the current position of the agent, v denotes candidate of the target position, q denotes the other agent, and μ is the parameter. Due to these state selections, the target position as valid sub-goal is generated and sent to the lower layer.

3.4 Q-learning for lower layer

In the lower layer, appropriate selection of concrete action to reach the target position decided at the upper layer should be fulfilled through reinforcement learning process. It should be noted that states of the other hunter agents are unnecessary for the lower task. The input state of the rule consists of the target position and the current own position. At every step in learning trial, the learning of the lower layer is employed because we can interpret every agent movement as the movement to current position considered as the movement to virtual targeted position according to another viewpoint. In the lower layer, Q-Learning (Sutton & Barto, 1998; Watkins & Dayan, 1988) can be applied successfully because the process is typical Markov Decision Process. Q-Learning is realized as:

$$
Q(s_{e,t},a_{e,t},c) = Q(s_{e,t},a_{e,t},c) + \alpha\left(r_t + \gamma \max_{\eta} Q(s_{e,t},\eta,c) - Q(s_{e,t},a_{e,t},c)\right) \tag{4}
$$

where Q is Q-value, $s_{e,t}$ is the state vector of the agent e at t-th step, $a_{e,t}$ is action of the agent e at t-th step, c denotes the state for updating, r denotes the reward, and a, γ are parameters. It should be noted that the current state of the agent moved from the other position always receive rewards considered as the virtual targeted state, internally.

4. A modular fuzzy model

4.1 Model structure

As a fuzzy model having high applicability, Single Input Rule Modules(SIRMs) (Seki et al., 2006; Yubazaki et al., 1997) was proposed. The idea is to unify reasoning outputs from fuzzy rule modules comprised with single input formed fuzzy if-then rules. The number of rules can be drastically reduced as well as bringing us high maintainability in actual application. However, its disadvantage of low precision is inevitable in order to apply the method to

huge multi-dimensional problems. I extend the SIRMs method by relaxing the restriction of the input space, i.e. single, to arbitrary subspace of the rule.
I propose a "Modular Fuzzy Model", for constructing the model of huge multi-dimensional space. Description of the model is as follows:

$$\begin{aligned} &Rules-1:\{if\ P_1(x)\ is\ A_j^1\ then\ y_1 = f_j^1(P_1(x))\}_{j=1}^{m_1} \\ &\qquad\vdots \\ &Rules-i:\{if\ P_i(x)\ is\ A_j^i\ then\ y_i = f_j^i(P_i(x))\}_{j=1}^{m_i} \\ &\qquad\vdots \\ &Rules-n:\{if\ P_n(x)\ is\ A_j^n\ then\ y_n = f_j^n(P_n(x))\}_{j=1}^{m_n} \end{aligned} \tag{5}$$

where "*Rules-i*" stands for the *i*-th fuzzy rule module, $P_i(x)$ denotes predetermined projection of the input vector x in *i*-th module, y_i is the output variable, and n is the number of rule modules. The number of constituent rules in the *i*-th fuzzy rule module is m_i. f is the function of consequent part of the rule like TSK-fuzzy model (Takagi & Sugeno, 1985). A_j^i denotes the fuzzy sets defined in the projected space.
The membership degree of the antecedent part of *j*-th rule in "*Rules-i*" module is calculated as:

$$h_j^i = A_j^i(P_i(x^0)) \tag{6}$$

where h denotes the membership degree and x^0 is an input vector. The output of fuzzy reasoning of each module is decided as the following equation.

$$y_i^0 = \frac{\sum_{k=1}^{m_i} h_k^i \cdot f_k^i(P_i(x^0))}{\sum_{k=1}^{m_i} h_k^i} \tag{7}$$

The final output of the "Modular Fuzzy Model" is formulated as:

$$y^0 = \sum_{i=1}^{n} w_i \cdot y_i^0 \tag{8}$$

where w_i denotes the parameter of importance of the *i*-th rule module. The parameter can be also formulated as the output of rule based system like modular neural network structure (Auda & Kamel, 1999). Figure 7 shows the structure of Modular Fuzzy Model.

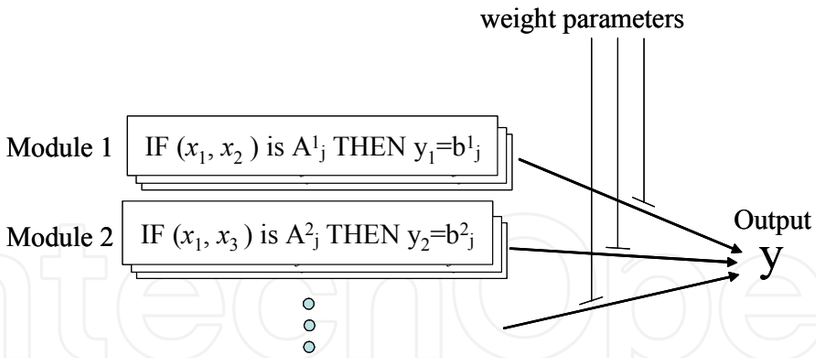


Fig. 7. Modular Fuzzy Model

4.2 Application of modular fuzzy model for upper layer

I tackle to the "curse of dimensionality" in the multi-agent pursuit problem using above proposed modular fuzzy model method. The objective of this study is to restrain memory consumption of rules in reinforcement learning keeping its performance. In this study, the function of consequent part in Eq.(5) is defined as parameter of "real value", i.e. simplified fuzzy reasoning model (Ichihashi & Watanabe, 1990), in order for applying to the pursuit problem as:

$$\begin{aligned} &Rules-1:\{if\ P_1(x)\ is\ A^1_j\ then\ y_1=b^1_j\}^{m_1}_{j=1} \\ &\qquad\vdots \\ &Rules-i:\{if\ P_i(x)\ is\ A^i_j\ then\ y_i=b^i_j\}^{m_i}_{j=1} \\ &\qquad\vdots \\ &Rules-n:\{if\ P_n(x)\ is\ A^n_j\ then\ y_n=b^n_j\}^{m_n}_{j=1} \end{aligned} \tag{9}$$

The importance parameter in Eq.(8) is set as 1.0 in this study. Instead of "crisp type" modular model described in section 3.3, I apply the modular fuzzy model to the upper layer model in the hierarchical reinforcement learning for pursuit problem. In addition to the usual crisp partition of the agent position as shown in Fig.8, fuzzy sets of the position are defined as shown in Fig.9. The antecedent fuzzy sets are defined by Cartesian products of each fuzzy set on the state of the agent position.

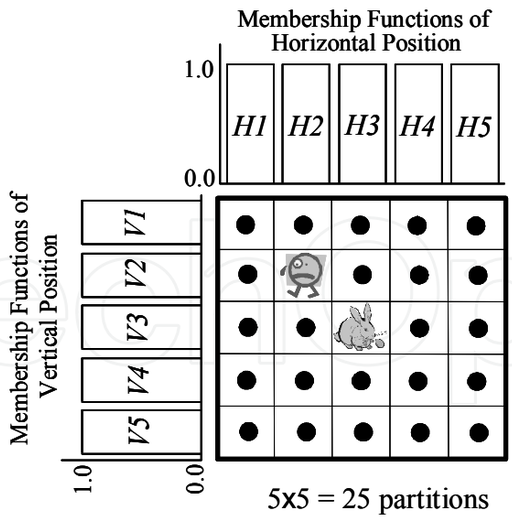


Fig. 8. Usual Crisp Partition of Agent Position

u in Eq.(2) is calculated by the modular fuzzy model and is learned considering the membership degree of the rules by the profit sharing algorithm. In this study, I assume that the number of fuzzy sets and parameters in the premise part is decided in advance. The parameters of real value in the consequent part are learned by the profit-sharing algorithm. The parameters are modified as:

$$\Delta b_j^i = \frac{h_j^i}{\sum_{k=1}^{m_i} h_k^i} k \tag{10}$$

where k denotes the reinforcement function in Eq.(2). The denominator in Eq.(10) can be omitted in actual processing because its value is always 1.0 from the definition of fuzzy sets described above.

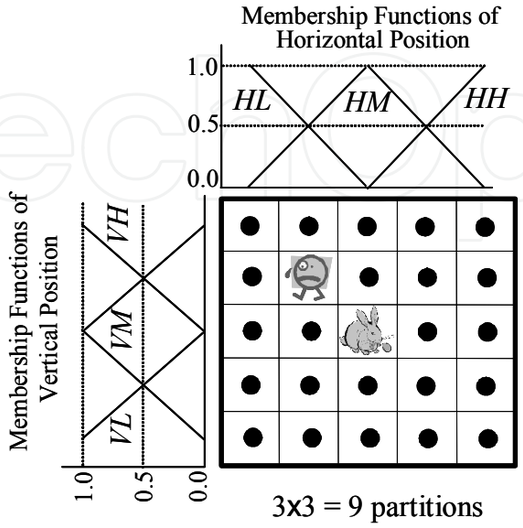


Fig. 9. Fuzzy Partition of Agent Position

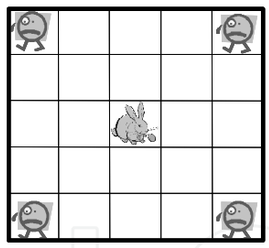


Fig. 10. Initial Placement of the Agents in 5x5 environment

5. Numerical experiments

5.1 Results compared with conventional learning methods

In the pursuit problem, the performance of the proposed hierarchical modular reinforcement learning method is compared with conventional methods through computer simulations. The size of the pursuit problem is 5x5. The absolute coordinate of the agent position is used in the experiments. The reason why relative coordinate is not used in the experiments is to evaluate essential performance of the proposed algorithm in terms of precision of learning, learning speed, and the memory consumption. As basic simulation conditions, each agent cannot communicate each other but can monitor the position of the other agents. The rule of the prey agent behavior is set as random behavior because the random behavior theoretically involves every action strategies. The initial placement of the prey agent and the hunter agents is shown in Fig.10.

The proposed methods are compared with the simple Q-Learning algorithm in order to evaluate basic performance of the methods. In the experiments, it is assumed that the Q-Learning agent(not hierarchically structured) can only utilize the position of the prey agent in addition to own position. The Q-Learning agent decides the action by calculating Q-value defined as $Q(g, s_e, a_e)$ from the sensed position of the prey agent and own position, where s_e is the position of the agent e, a_e is the corresponding action of the agent e, and g is the position of the prey agent.

As for hierarchical modular reinforcement learning agents, three methods are simulated. The expressions of the upper layer are different, though their hierarchical structures and the lower layer driven by Q-Learning are the same. The first method is structured as the complete expressed upper layer. From all positions of the hunter agents and the prey agent, the target position to move is decided. The number of rules in upper layer is 25*25*25*25*25=9,765,625. The second method is "crisp" modular model for upper layer. The number of rules in upper layer of each agent is (25*25*25*25)*3= 1,171,875. The last method is the modular fuzzy model for upper layer. Detailed constructions of the model are described in next subsection. For example, the 1st agent of the modular fuzzy model for upper layer is constructed as:

$$\begin{aligned} Rules-1: &\ \{if\ [g,h_1,h_2,h_3]\ is\ A_j^1\ then\ \ y_1=b_j^1\}_{j=1}^{50,625} \\ Rules-2: &\ \{if\ [g,h_1,h_2,h_4]\ is\ A_j^2\ then\ \ y_2=b_j^2\}_{j=1}^{50,625} \\ Rules-3: &\ \{if\ [g,h_1,h_3,h_4]\ is\ A_j^3\ then\ \ y_3=b_j^3\}_{j=1}^{50,625} \end{aligned} \tag{11}$$

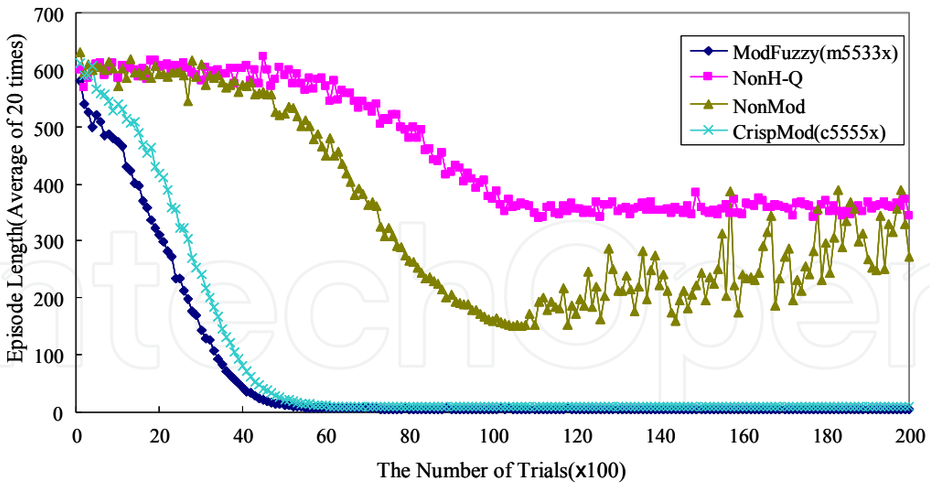


Fig. 11. Simulation Results

where g is the position of the prey agent, h is the position of the hunter agent, and b is the parameter of consequent part of the fuzzy rule. The fuzzy set A is constructed by combining the crisp sets of own agent position and prey agent position with the fuzzy sets of the other two hunter agent positions defined by partitioning the grid into 3×3 as shown in Fig.9. The number of rules in upper layer is much smaller than the others, i.e. (25*25*9*9)*3=151,875.

I perform the simulation 20 times for each method. The number of trials in the simulation are 20,000. The results are shown in Fig.11. The depicted data is averaged value of 20 series after averaging each sequential 100 trials. The results by the modular fuzzy model(depicted as ModFuzzy) show the best performance compared with the other methods. Both the learning speed and the precision of learning are desirable. Furthermore required memory amount is much smaller than the other methods. The results by "crisp" modular model(depicted as CrispMod) show also good performance. The complete expression model(depicted as NonMod) cannot acquire rules efficiently and the performance is deteriorated over time. This seems to be caused by the sparsity of model expression. The simple Q-Learning agent (NonH-Q) is not so bad unexpectedly in the small 5×5 grid world. The strategy only to approach to the prey agent acquired by the simple non-hierarchical Q-Learning might be reasonable in such small world. However, as the knowlede about surrounding task cannot be learned at all in such model expression, successful surrounding completely depends upon accidental behavior of the prey agent.

5.2 Detailed results by proposed model

In order to construct the modular fuzzy model, the important issue is to decide the dimension of projection in rule modules. Furthermore the number of partition should be also decided appropriately. In the pursuit problem, as the positions of own agent and the prey agent are indispensable by nature, the issue is restricted to decide the number of the other hunter agents included in model expression and the number of partition, i.e. crisp or fuzzy. In this study, the projection is extended step by step through modeling(reinforcement learning) from one other hunter agent added. The number of partition for each position is

changed as well as the dimension. The results are summarized in Table 1. In this Table, averaged value, standard deviation, and standard error of episode lengh average of last 100 trials in 20 times simulation are shown as well as the number of partition and the number of

Model ID	The Number of Partition of Agent Position					The Number of Rules for One Agent	Episode Length of Last 100 trials (20 times)		
	Target	Own	Other1	Other2	Other3		Average	Standard Deviation	Standard Error
m333xx	9	9	9			2,187	225.77	310.71	69.48
m533xx	25	9	9			6,075	142.76	68.08	15.22
m335xx	9	9	25			6,075	98.27	44.75	10.01
m353xx	9	25	9			6,075	8.25	1.70	0.38
m535xx	25	9	25			16,875	121.99	85.53	19.12
m553xx	25	25	9			16,875	5.97	0.50	0.11
m355xx	9	25	25			16,875	10.94	1.06	0.24
c555xx	25	25	25			46,875	11.30	22.20	4.96
m3355x	9	9	25	25		151,875	115.76	33.90	6.92
m5533x	25	25	9	9		151,875	5.81	0.33	0.07
c5555x	25	25	25	25		1,171,875	9.07	0.67	0.14
u55555	25	25	25	25	25	9,765,625	271.49	283.88	63.48

Notes of Model ID: m5533x

- The number of partition: Target, Own, Other1, Other 2, Other3
  - 3 : fuzzy partition
  - 5 : crisp partition
  - x : void (not used in model)
- m : modular fuzzy model
- c : crisp modular model
- u : usual memory type

Table 1. Detailed Results of Modular Model

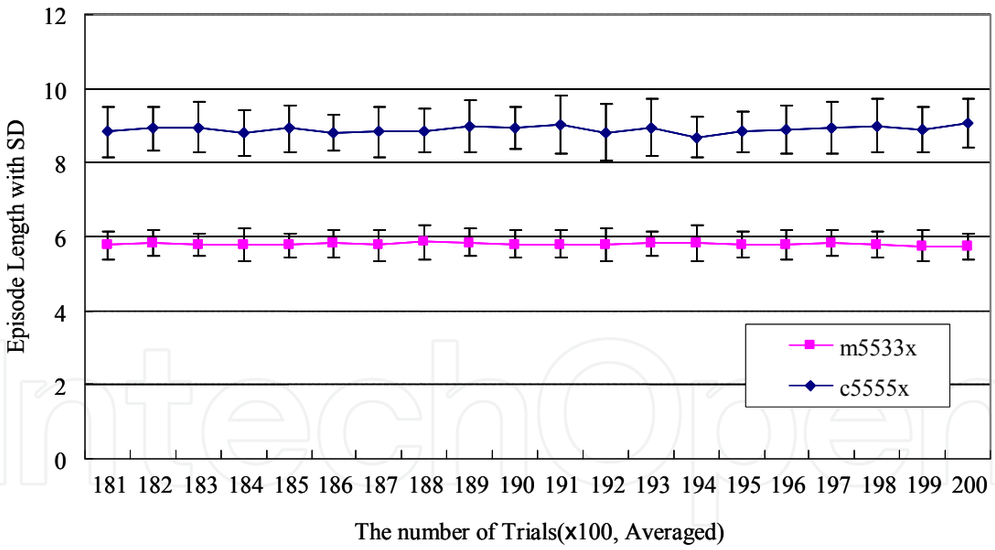


Fig. 12. Comparison of Modular Fuzzy Model and Crisp Modular Model

rules corresponding to the model. From the results of first four models, own position of the agent might be partitioned by crisp sets, i.e. m353xx. From further results of next four models, own position of the agent and position of the target, i.e. prey agent, might be partitioned by crisp sets, i.e. m553xx. From these obserbations, the model construction is heuristically performed as shown in the last four results in the Table. From the results m5533x model has best performance among the models. Compared results with good

model(c5555x) are shown in Fig.12. The significance of the m5533x model performance compared with the other good model performance is also investigated by the *t* test. The result compared with m553xx model is that null hypothesis, i.e. the means do not differ, is rejected with statistical significance level of 0.01. As the results compared with the other model are obvious, the description is omitted.

The results by the proposed model are considered that the learned agent can perform surroundig task within six times movement against almost all behavior pattern of the prey agent. This level cannot be attained without collaborative behavior of the learned agent. In addition to its drastically improved learning speed, it can be said that the precision level of learning is sufficient compared with the conventional techniques.

6. Conclusion

In this chapter, I focused on the pursuit problem and proposed a hierarchical modular reinforcement learning that Profit Sharing learning algorithm is combined with Q Learning reinforcement learning algorithm hierarchically in multi-agent environment. As the model structure for such huge problem, I proposed a modular fuzzy model extending SIRMs architecture. Through numerical experiments, I showed the effectiveness of the proposed algorithm compared with the conventional algorithms. My future plan concerning with the proposed methods includes application of another multi-agent problem or complex task problem.

7. References

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning*, MIT Press

Watkins, C. J. & Dayan Y. (1988). Technical Note: Q-Leaning, *Machine Learning*, Vol.8, pp.58-68

Grefenstette, J. J. (1988). Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms, *Machine Learning*, Vol.3, pp.225-245

Miyazaki. K.; Kimura. H. & Kobayashi. S. (1999). Theory and Application of Reinforcement Learning Based on Profit Sharing, *Journal of JSAI*, Vol.14, No.5, pp. 800-807

Miyazaki. S.; Arai. S. & Kobayashi. S. (1999). A Theory of Profit Sharing in Multi-agent Reinforcement Learning, *Journal of JSAI*, Vol. 14, No.6, pp.1156-1164

Benda. M.; Jagannathan. V. & Dodhiawalla. R. (1985). On Optimal Cooperation of Knowledge Sources, *Technical Report*, BCS-G2010-28, Boeing AI Center

Ito. A. & Kanabuchi. M. (2001). Speeding up Multi-Agent Reinforcement Learning by Coarse-Graining of Perception –Hunter Game as an Example-, *Transaction of IEICE*, Vol.J84-D-1, No.3, pp.285-293

Takahashi. Y. & Asada. M. (1999). Behavior Acquisition by Multi-Layered Reinforcement Learning, *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics.*, pp.716-721

Morimoto. J. & Doya. K. (2000). Acquisition of Stand-up Behavior by a Real Robot using Hierarchical Reinforcement Learning, *Proceedings of International Conference on Machine Learning*, pp. 623-630

Ono. N. & Fukumoto. K. (1996). Multi-agent Reinforcement Learning: A Modular Approach, *Proceedings 2nd International Conference on Multi-agent Systems*, pp.252-258, AAAI Press

Fujita. K. & Matsuno. H. (2005). Multi-agent Reinforcement Learning with the Partly High-Dimensional State Space, *Transaction of IEICE*, Vol.J88-D-1, No.4, pp.864-872

Murano. H. & Kitamura. S. (1997). Q-Learning with Adaptive State Segmentation(QLASS), *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp.179-184

Hamagami. T.; Koakutsu. S. & Hirata. H. (2003). An Adjustment Method of the Number of States on Q-Learning Segmenting State Space Adaptively, *Transaction of IEICE*, Vol. J86-D1, No.7, pp.490-499

Seki. H.; Ishii. H. & Mizumoto. M. (2006). On the Generalization of Single Input Rule Modules Connected Type Fuzzy Reasoning Method, *Proceedings of the SCIS&ISIS2006*, pp.30-34

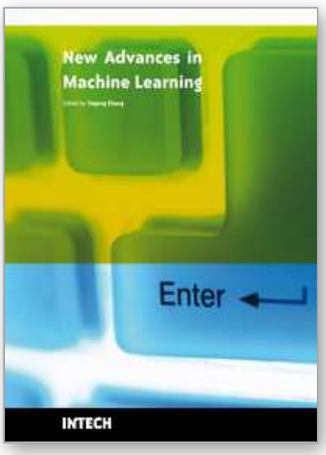
Yubazaki. N.; Yi. J.; Otani. M. & Hirota. K. (1997). SIRMs Dynamically Connected Fuzzy Inference Model and Its Applications, *Proceedings of IFSA'97*, vol.3, pp.410-415

Takahashi. Y. & Watanabe. T. (2006). Learning of Agent Behavior Based on Hierarchical Modular Reinforcement Learning, *Proceedings of the SCIS&ISIS2006*, pp.90-94

Ichihashi. H. & Watanabe. T. (1990). Learning Control System by a Simplified Fuzzy Reasoning Model, *Proceedings of the 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp.417-419

Takagi. T. & Sugeno. M. (1985). Fuzzy Identification of Systems and Its Applications to Modeling and Control, *IEEE Transaction on Systems, Man, and Cybernetics*, Vol. 15, pp. 116-132

Auda. G. & Kamel. M. (1999). Modular Neural Networks: A Survey, *International Journal of Neural Systems*, Vol.9, No.2, pp.129-151



New Advances in Machine Learning

Edited by Yagang Zhang

ISBN 978-953-307-034-6
Hard cover, 366 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The purpose of this book is to provide an up-to-date and systematical introduction to the principles and algorithms of machine learning. The definition of learning is broad enough to include most tasks that we commonly call “learning” tasks, as we use the word in daily life. It is also broad enough to encompass computers that improve from experience in quite straightforward ways. The book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides a good introduction to many approaches of machine learning, and it is also the source of useful bibliographical information.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen