International Conference on Control, Automation and Systems 2010
Oct. 27-30, 2010 in KINTEX, Gyeonggi-do, Korea

An Intelligent Control System Construction Using High-level Time Petri Net And Reinforcement Learning

Liangbing Feng[1], Masanao Obayashi[1], Takashi Kuremoto[1] and Kunikazu Kobayashi[1]

[1] Division of Computer Science & Design Engineering, Yamaguchi University, Ube, Japan
(Fax: +81-836-85-9501, E-mail: n007we @yamaguchi-u.ac.jp)

Abstract: A hybrid intelligent control system model which combines high-level time Petri net (HLTPN) and Reinforcement Learning (RL) is proposed. In this model, the control system is modeled by HLTPN and system state last time is presented as transitions delay time. For optimizing the transition delay time through learning, a value item is appended to delay time of transition for recording the reward from environment and this value is learned using Q-learning – a kind of RL. Because delay time of transition is continuous, two RL algorithms in continuous space methods are used in Petri Net learning process. Finally, for the purpose of certification of the effectiveness of our proposed system, it is used to model a guide dog robot system which system environment is constructed using radio-frequency identification (RFID). The result of the experiment shows the proposed method is useful and effective.

Keywords: Intelligent control, Reinforcement learning, Petri net, RFID

1. INTRODUCTION

Petri net combines a well-defined mathematical theory with a graphical representation of the dynamic behavior of the system. The theoretic aspect of Petri nets allows precise modeling and analysis of system behavior, while the graphical representation of Petri nets enable visualization of state changes of the modeled system [1]. The Petri net is therefore widely used to model various control systems. But the traditional Petri net doesn't have learning capability, all the parameters which describe the system characteristics need to be set individually and empirically when a dynamic system is modeled. On the other hand, Reinforcement Learning is a framework for an agent to learn the choice of an optimal action based on a reinforcement signal [2]. Recently, there are some researches for making the Petri net is equipped with learning capability. A learning Petri net model which combines Petri net with neural network is proposed in [3]. This learning Petri net model can realize an input-output mapping through Petri net's weight function which is adjusted just like an artificial neural network. And the learning method was applied to the nonlinear system control. In paper [4], a Petri net model which combines the Petri net and reinforcement learning is also proposed. This model can adjust the arc weight function using Q-learning and is used to model a robot system. The model makes system have capability to optimize its behavior through interaction with environment when dynamic system is running.

In this paper, we propose an intelligent control system model, in which system is specified by a High-level Petri net and transition delay time is learned using reinforcement learning. For solving continuous delay time learning, two continuous space learning methods are used.

2. PROPOSED SYSTEM

2.1 Architecture

The intelligent control system is constructed by three sectors: Petri net control model, Action controller and Delay time evaluator. Figure 1 shows its overall architecture.

Petri Net control model: The control system is modeled by a High-level time Petri net (HLTPN). Petri net's place, transition, and colored token present system state, action, and control signal, respectively. Some transitions have delay time which presents system action delay time or pre-state last time.

Action controller: Action controller transfers the transition's fire of Petri net to system control signal.

Delay time Evaluator: It evaluates delay time according to the feedback signal from environment and uses this evaluated value to adjust the delay time of Petri net model. This is learning part of the system. It makes the proposed system have learning capability.

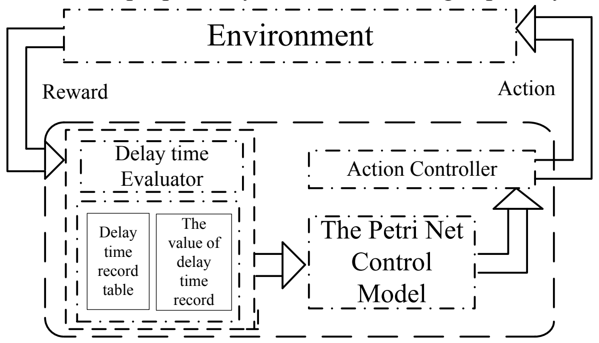


Fig. 1 Proposed system

2.2 Definition of HLTPN and its extension

The proposed system is modeled by a High-level time Petri net (HLTPN). HLTPN is an expanded Petri net, in which tokens are differentiated by colors, its enabled transitions have a time delay and arcs are restrained by weight functions which are expressed as the number and color of Tokens that are consumed or generated when a transition fired [5].

Definition 1: HLTPN has a structure $HLTPN=(NG, C, W, DT, M_0)$ where

(i) $NG=(P, T, F)$ is called net graph with P a finite set of nodes, called Places. ID: $P\rightarrow N$, is a function marking P, $N = (1, 2, ...)$ is the set of natural number. Using $p_1, p_2, ..., p_n$ represents the elements of P and n is the cardinality of set P;

T a finite set of nodes, called Transitions, which disjoint from P, $P \cap T = \emptyset$; $ID{:}T \rightarrow N$ is a function marking T. Using $t_1, t_2, ..., t_m$ represents the elements of T, m is the cardinality of set T;

$F \subseteq (P \times T) \cup (T \times P)$ is a finite set of directional arcs, known as the flow relation;

(ii) C is a finite and non-empty color set for describing difference type data;

(iii) $W{:}\ F \rightarrow C$ is a weight function on F. If $F \subseteq (P \times T)$, the weight function W is W_{in} which defines colored Token can through the arc and enable a T. Those color token will be consumed when transition fire. If $F \subseteq (T \times P)$, the weight function W is W_{out} which defines colored Token will be generate by T and be input the P.

(iv) $DT{:}\ T \rightarrow N$ is a delay time function of a transition which has a T delay for an enable transition fired.

(v) M_0: $P \rightarrow \cup_{p \in P} \mu C(p)$ such that $\forall p \in P$, $M_0(p) \in \mu C(p)$ is the initial marking function which associates a multi-set of tokens of correct type with each place.

For using RL to adjust the parameter of LTPN, HLTPN will be extended.

Definition 2: extend HLTPN has a structure, ex-$HLTPN = (HLTPN, VT)$, where

(i) $HLTPN = (NG, C, W, DT, M_0)$ is a High-Level Time Petri Net.

(ii) VT (the value of delay)$:\ DT \rightarrow R$, is a function marking DT. Here every transition delay time DT has a reward value $R \in$ real number.

2.3 Learning algorithm

The delay time of transition is a continuous variable. So, the delay time learning is a problem of RL in continuous action spaces. Now, there are several methods of RL in continuous spaces: discretization method, function approximation method, and so on [6]. Here, the Discretization method and function approximation method are used in the delay time learning.

1) Discretization method:

As shown in Fig. 2 (1), transition T_1 has a delay time t_1. When P_1 has Token <$Token_n$>, the system is at a state which P_1 has a Token. This time transition T_1 is enabled. Because T_1 has a delay time t_1, T_1 don't fire immediately. After lasting time t_1 and T_1 fires, the Token in P_1 is taken out and this state is terminated. Then, delay time of T_1 is P_1 state lasting time.

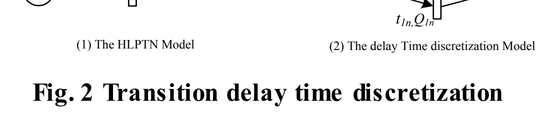


Fig. 2 Transition delay time discretization

Because the delay time is a continuous variable, we discretize the different delay time for using reinforcement learning to optimize delay time. For example, T_1 in Fig. 2 (1) has an undefined delay time t_1. T_1 is discretized into several different transitions which have different delay time (shown in Fig. 2 (2)) and every delay time has a value item Q. After T_1 fired at delay time t_{1i}, it gets a reward r immediately or after its subsequence gets rewards. The value of Q is updated by formula (1).

$$Q(P,T) \leftarrow Q(P,T) + \alpha\,[r + \gamma Q(P',T') - Q(P,T)]. \qquad (1)$$

where, $Q(P, T)$ is the value of transition T at Petri net state P. $Q(P',T')$ is the value of transition T' at next state P' of P. α is the step-size, γ is a discount rate.

After renewing of Q, the optimal delay time will be selected. In Fig. 2 (2), when $T_{11} \ldots T_{1n}$ get value $Q_{11} \ldots Q_{1n}$, the transition is selected by the soft-max method according to a probability of Gibbs distribution.

$$\Pr\{t_t = t | p_t = p\} = \frac{e^{Q(p,t)\beta}}{\sum_{b \in A} e^{Q(p,b)\beta}}. \qquad (2)$$

where, $\Pr\{t_t = t | p_t = p\}$ is a probability selecting transition t at state p, β is a positive inverse temperature constant and A is a set of available transitions.

```
Transition's delay time learning algorithm 1 (Discretization method):
Step 1. Initialization: discretize delay time and set
        $Q(p,t)$ of every transition's delay time to zero.
Step 2. Initialize Petri net, i.e. make the Petri net state
        as $P_1$.
    Repeat (i) and (ii) until system becomes end state.
      (i) Select a transition using formula (2).
      (ii) After transition fired and reward is observed
           value of $Q(p,t)$ is adjusted using formula (1).
Step 3. Repeat step2 until $t$ is optimal as require.
```

2) Function approximation method

First, the transition delay time is selected randomly and executed. The value of delay time is obtained using formula (1). When the system is executed m times, the data $(t_i, Q_i(p,t_i))$ $(i = 1, 2, \ldots, m)$ is yielded.

The relation of value of delay time Q and delay time t is supposed as $Q = F(t)$. Using least squares method, $F(t)$ will be obtained as follows.

It is supposed that F is a function class which is constituted by a polynomial. And it is supposed that formula (3) hold.

$$f(t) = \sum_{k=0}^{n} a_k t^k \in F. \qquad (3)$$

The data $(t_i, Q_i(p,t_i))$ are substituted in formula (3). Then:

$$f(t_i) = \sum_{k=0}^{n} a_k t_i^k \quad (i = 1, 2, \ldots, m;\ m \geq n). \qquad (4)$$

Here, the degree m of data $(t_i, Q_i(p,t_i))$ is not less than data number n of formula (3). According to least squares method, we have (5).

$$\|\delta\|^2=\sum_{i=1}^{m}\delta_i^2=\sum_{i=1}^{m}[\sum_{k=0}^{n}a_k t_i^k - Q_i]^2 \Rightarrow \min . \qquad (5)$$

In fact, (5) is a problem which evaluates the minimum solution of function (6).

$$\|\delta\|^2=\sum_{i=1}^{m}[\sum_{k=0}^{n}a_k t_i^k - Q_i]^2 \cdot \qquad (6)$$

So, function (7), (8) are gotten from (6).

$$\frac{\partial\|\delta\|^2}{\partial a_j}=2\sum_{i=1}^{m}\sum_{k=0}^{n}(a_k t_i^k - Q_i)t_i^j=0 \; (j=0,\ 1,\ \ldots,\ n), \qquad (7)$$

$$\sum_{i=1}^{m}(\sum_{k=0}^{n}t_i^{j+k})a_k=\sum_{i=1}^{m}t_i^j Q_i \; (j=0,\ 1,\ \ldots,\ n). \qquad (8)$$

Solution of Equation (8) a_0, a_1, …, a_n can be deduced and $Q = f(t)$ is attained. The solution t^*_{opt} of $Q = f(t)$ which makes maximum Q is the expected optimal delay time.

$$\frac{\partial f(t)}{\partial t}=0. \qquad (9)$$

The multi-solution of (9) $t = t_{\text{opt}}$ $(opt = 1, 2, \ldots, n\text{-}1)$ is checked by function (3) and a $t^*_{opt} \in t_{\text{opt}}$ which makes $f(t^*_{opt})= \max f(t_{\text{opt}})$ $(opt = 1, 2, \ldots, n\text{-}1)$ is the expected optimal delay time. We use t^*_{opt} as delay time and execute the system and get new $Q(p,\ t^*_{opt})$. This $(t^*_{opt},\ Q(p,\ t^*_{opt}))$ is used as new and the least squares method can be used again to acquire more precise delay time.

Transition's delay time learning algorithm 2 (Function approximation method):

Step 1. Initialization: Set $Q(p,\ t)$ of every transition's delay time to zero.

Step 2. Initialize Petri net, i.e. make the Petri net state as P_1.

Repeat (i) and (ii) until system becomes end state.

(i) Randomly select the transition delay time t.

(ii) After transition fires and reward is observed, value of $Q(p,\ t)$ is adjusted using formula (1).

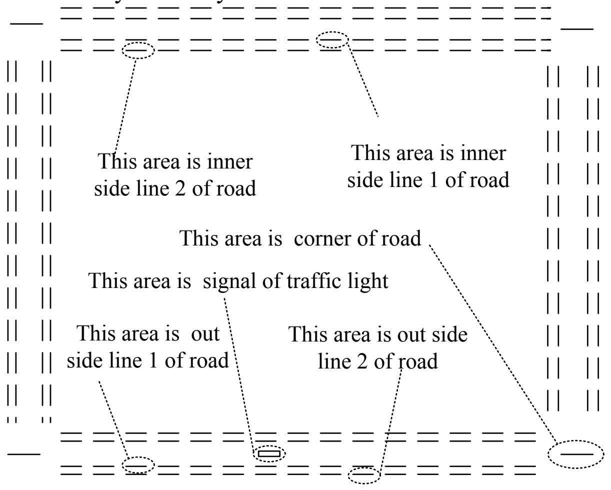
Step 3. Repeat step 2 until got adequacy data. Then, Evaluate the optimal t using function approximation method.

3. APPLICATION TO RFID NAVIGATION GUIDE DOG ROBOT SYSTEM

The proposed system was applied to guide robot system which uses RFID (Radio-frequency identification) to construct experiment environment. The RFID is used as navigation equipment for robot motion. The performance of the proposed system is evaluated through computer simulation and real robot experiment.

3.1 The RFID environment construction

RFID tags are used to construct a blind road which showed in Fig. 3. There are forthright road, corner and traffic light signal areas. The forthright road has two group tags which have two lines RFID tags. Every tag is stored with the information about the road. The guide dog robot moves, turns or stops on the road according to the information of tags. For example, if the guide dog robot reads corner RFID tag, then it will turn on the corner. If the guide dog robot reads either outer or inner side RFID tag, it implies that the robot will deviate from the path and robot motion direction needs adjusting. If the guide dog robot reads traffic control RFID tag, then it will stop or run unceasingly as traffic light signal which is dynamically written to RFID.



This area is inner side line 2 of road
This area is inner side line 1 of road
This area is corner of road
This area is signal of traffic light
This area is out side line 1 of road
This area is out side line 2 of road

Fig. 3 RFID construction environment

3.2 The Petri net model for the guide dog

The extend HLTPN control model for guide dog robot system is presented in Fig. 4. The mining of place and transition is listed below:

P1	System start state	P2	Getting RFID information
P3	Turn corner state	P4	Left adjust state
P5	Right adjust state	T1	Read RFID environment
T2	Stop guide dog	T3	Guide dog runs
T4	Start turn corner state	T5	Start left adjust state
T6	Start right adjust state	T7	Stop turn corner state
T8	Stop left adjust state	T9	Stop right adjust state

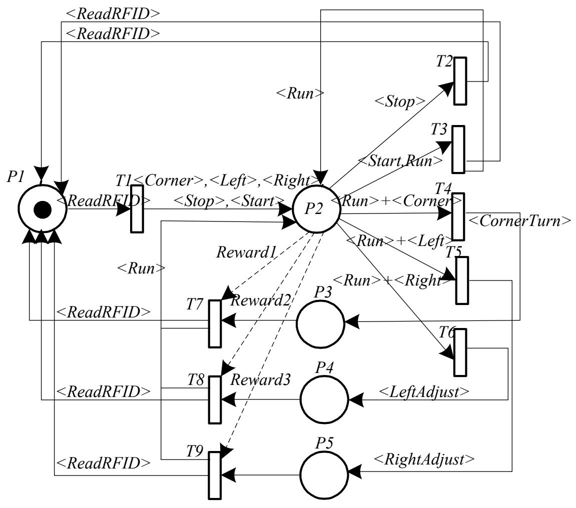


Fig. 4 Learning Petri net model for Guide dog

When the system begins running, it firstly reads

RFID environment and gets the information Token putting P_2. These Tokens fire one of transition from T_2 to T_6 according to weight function on P_2 to T_2, …, T_6. Then, the guide dog enters stop, running, turning corner, left adjust or right adjust states. Here, at P_3, P_4, P_5 states, the guide dog turns at a specific speed. The delay time of T_7-T_9 decide the correction of guide dog adjusting its motion direction. This delay time is learned through reinforcement learning algorithm which described in Section 2.3.

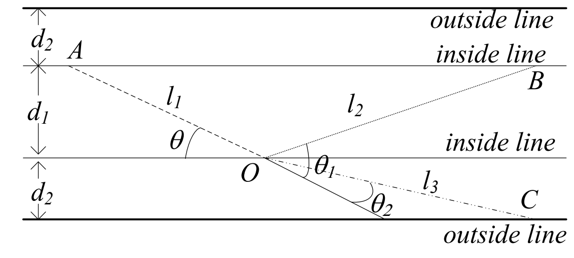
3.3 The reward getting from environment

When T_7, T_8 or T_9 fires, it will get reward r as formula (10) (b) when the guide dog doesn't get Token <Left> and <Right> until getting Token <corner> i.e. the robot runs according correct direction until arriving corner. It will get reward r as formula (10) (a), where t is time from transition fire to get Token <Left> and <Right>. On the contrary, it will get punishment -1 as (10) (c) if robot runs out the road.

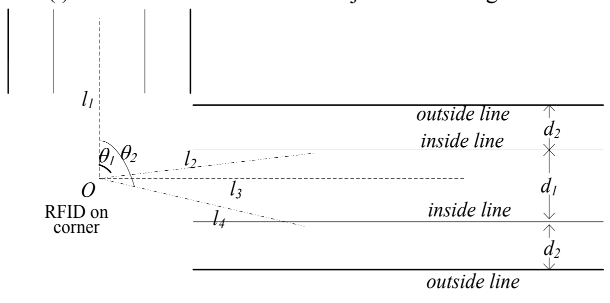
$$r=\begin{cases} 1/e^t & (a) \\ 1 & (b) \\ -1 & (c) \end{cases}. \qquad (10)$$

3.4 Computer simulation and real robot experiment

When robot reads the <Left>, <Right> and <corner> information, it must adjust the direction of motion. The amount of adjusting is decided by last time of robot at state of P_3, P_4 and P_5. So, the delay time of T_7, T_8 and T_9 need to learn.



(i) Direction of robot motion adjust on forthright road



(ii) Direction of robot motion adjust at corner

Fig. 5 Direction of robot motion adjust

Before the simulation, some robot motion parameter symbols are given as:

v	velocity of robot
ω	angular velocity of robot
t_{pre}	last time of former state
t	the adjust time
t_{post}	last time of the state after adjusting

v, ω, t_{pre}, t_{post} can be measure by system when robot is running. The delay time of T_7, T_8 and T_9, i.e. robot motion adjust time, is simulated in two cases.

1) As shown in Fig. 5 (i), when robot is running on forthright road and meets inside RFID line, its deviation angle θ is:

$$\theta = \arcsin(d_1/l_1) = \arcsin(d_1/(t_{pre}\bullet v)). \qquad (11)$$

where d_1 and l_1 are width of area between two inside lines and motion length between two time read RFID respectively (See Fig. 5).

Robot's adjust time (transition delay time) is t. If $t\bullet\omega-\theta\geq 0$, then

$$t_{post} = \frac{d_1}{v\sin(t\omega-\theta)}, \qquad (12)$$

else

$$t_{post} = \frac{d_2}{v\sin(\theta-t\omega)}. \qquad (13)$$

Here, t_{post} is used to calculate reward r using formula (10). In the same way, the reward r can be calculated when robot meets outside RFID line.

When robot is running on forthright road and meets outside RFID line, the deviation angle θ is

$$\theta = \arcsin(d_2/(t_{pre}\bullet v)), \qquad (14)$$

Robot's adjust time (transition delay time) is t. If $t\bullet\omega-\theta\geq 0$, then

$$t_{post} = \frac{d_2}{v\sin(t\omega-\theta}, \qquad (15)$$

else robot will runs out the road. And the reward r is calculated using formula (10).

2) As shown in Fig. 5 (2), when robot is running at corner, it must adjust θ=90°. If $\theta\neq$90°, robot will read <Left>, <Right> after it turn corner. Now, the case which robot will read inner line <Left>, <Right> will be considered. If robot's adjusting time is t. If $t\bullet\omega-\theta\geq0$, then

$$t_{post} = \frac{d_1}{2v\sin(t\omega-\theta)}, \qquad (16)$$

else

$$t_{post} = \frac{d_1}{2v\sin(\theta-t\omega)}. \qquad (17)$$

Same to case (1), t_{post} is used to calculate reward r using formula (9). In the same way, the reward r can calculate when robot meets outside RFID line.

The calculation of reward t for other cases of robot direction adjusting is considered as the above two cases.

In this simulation, the value of delay time has only a maximum at optimal delay time point. The graph of relation for delay time and its value is parabola. So, when transition's delay time learning algorithm 2 (function approximation method) is used, the relation of delay time and its value is assumed as:

$$Q = a_2t^2 + a_1t + a_0. \qquad (18)$$

Computer simulations of Transition's delay time learning algorithm 1 and 2 were executed in the all cases of robot direction adjusting. In the simulation of algorithm 1, the positive inverse temperature constant β is set as 10.0. After the delay time of different cases was learned, it is recorded in a delay time table. Then, real robot experiment was carried out using the delay time table which was obtained by simulation process. The real experimental environment is shown in Fig. 6.

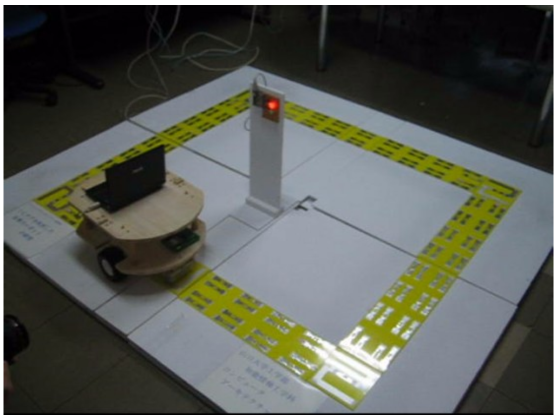
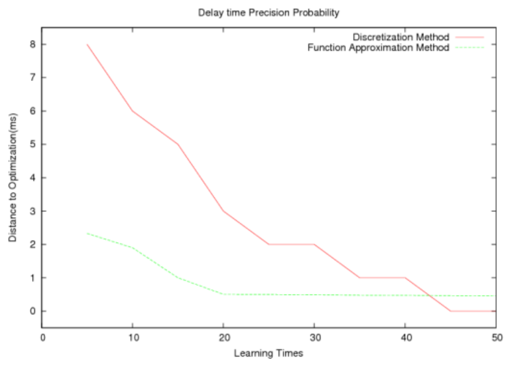


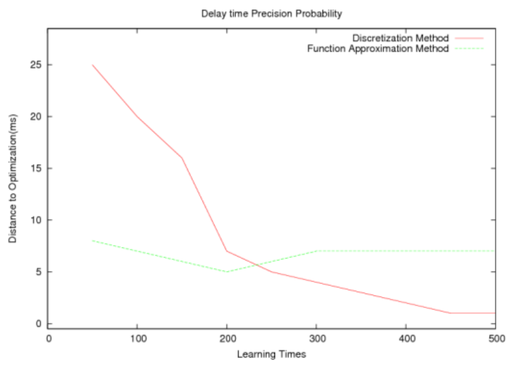
Fig. 6 The real experimental environment

3.5 The result of simulation and experiment

The simulation result of transition's delay time learning algorithm 1, 2 in two cases is shown in Fig. 7.



(i) Simulation result of motion adjusting on forthright road



(ii) Simulation result of motion adjusting at corner
Fig. 7 The result of simulation

The simulation result of θ=5° when robot motion adjusting on forthright road is shown in Fig. 7 (i). The simulation result of robot motion adjust at corner is shown in Fig. 7 (ii). From the result, it is found that function approximation method can quickly approach optimal delay time than discretization method but discretization method can more nearly approach optimal delay time through long time learning.

4. CONCLUSIONS

We proposed a hybrid intelligent control system which combines a high-level time Petri net (HLTPN) and reinforcement learning (RL) in this paper. The system state last time is presented as transition delay time of HLTPN. For learning the transition delay time, a extend HLTPN is proposed. In the extend HLTPN, delay time is learned using reinforcement learning when system interacts with the environment. For solving continuous delay time learning, two continuous space learning methods are used in learning algorithm. Finally, the proposed system was used to model a guide dog robot system where the system environment was constructed using RFID. The result of experiment shows the proposed method is useful and effective. We plan to use RL algorithm to adjust other parameters of Petri net and extend our work to model other dynamic systems in the future.

REFERENCES

[1] J. Wang, "Petri nets for dynamic event-driven system modeling". *Handbook of Dynamic System Modeling*, Ed: Paul Fishwick, *CRC Press*, pp. 1-17, 2007.

[2] Richard. S. Sutton, A.G. Barto, "Reinforcement Learning", *The MIT Press*, 1998.

[3] Hirasawa K., Ohbayashi M., Sakai S., Hu J., "Learning Petri Network and Its Application to Nonlinear System Control", *IEEE Transactions on Systems, Man and Cybernetic, Part B: Cybernetics*, 28(6), pp. 781-789, 1998.

[4] Liangbing Feng., M. Obayashi, T. Kuremoto and K. Kobayashi, "A Learning Petri Net Model Based on Reinforcement Learning", *Proceedings of the 15th International Symposium on Artificial Life and Robotics (AROB2010)*, pp. 290-293, February 4-6, 2010.

[5] Guangming C., Minghong L., Xianghu W., "The Definition of Extended High-level Time Petri Nets", *Journal of Computer Science*, 2(2), pp. 127-143, 2006.

[6] Doya, K., " Reinforcement Learning in Continuous Time and Apace", *Neural Computation*, Vol.12, No. 219-245, 2000.