Reinforcement Learning Intersection Controller

Nurlan S. Dairbekov, Gulnur Tolebi, Daniyar Kurmankhojayev, and Ravil Mussabayev

Kazakh-British Technical University, Almaty, Kazakhstan,
tolebi.glr@gmail.com

Abstract. This paper presents an online model-free adaptive traffic signal controller for an isolated intersection using a Reinforcement Learning (RL) approach. We base our solution on the Q-learning algorithm with action-value approximation. In contrast with other studies in the field, we use the queue length in adddition to the average delay as a measure of performance. Also, the number of queuing vehicles and the green phase duration in four directions are aggregated to represent a state. The duration of phases is a precise value for the non-conflicting directions. Therefore, cycle length is non-fixed. Finally, we analyze and update the equilibrium and queue reduction terms in our previous equation of an immediate reward. Also, the delay based reward is tested in the given control system. The performance of the proposed method is compared with an optimal symmetric fixed signal plan.

Keywords: Intelligent Traffic Signal Controller, Intelligent Transportation System, Reinforcement Learning, Artificial Neural Network

1 Introduction

Our study aim to present a traffic control system on an isolated intersection. In the given work, one of the solutions - RL with action-value approximation (Deep Q-Network, DQN) considered to tackle the problem of traffic flow control on an isolated intersection. Our previous research [1] based on Q-table implementation of Q-learning algorithm for an intersection controller, where we define nine states with three actions that interatively refine the signal plan. The previous approach has a serious limitation, such as it can not easily incorporate data from different sources. For instance, week day, day time, temperature, season, and etc. The remedy is action-value function approximation by a neural network overcomes the issue. After that, we decided to change the action space such that instead of adding and subtraction from green phases dt we directly set their duration. In other words, new action space consists of a set of phase durations. That does not change the way how the system works, but that change conceptually means a lot. To be more specific, now it implicity takes into account the previous green phase durations since the information about lengths of the phases directly incorporated in the action itself. The limitations of our immediate reward formula were releaved and proposed the updated form. Therefore, each of the four directions is considered separately.

1.1 Reinforcement Learning

Reinforcement learning (RL) studies how to teach an agent to interact efficiently with the *environment* (Fig. 1). An *agent* is the learner which is able to sense the state of its environment and take actions that affect the state. On each consecutive interaction the environment responds by sending a numerical *reward signal* to the agent which indicates how good the action was. The agent must have a *goal* relating to the state of the environment. The purpose of reinforcement learning is for the agent to learn the action selection rule which ensures the maximum possible total reward the agent receives over the long run so as to achieve the goal.
The agent-environment interaction is continual and occurs over time t as follows:

$$S_t \underset{A_t}{\longrightarrow} (R_{t+1}, S_{t+1}), \tag{1}$$

where $S_t \in \mathcal{S}$ and $S_{t+1} \in \mathcal{S}$ are the states of the environment at the previous and current times respectively; $A_t \in \mathcal{A}$ is the action taken, and $R_{t+1} \in \mathcal{R}$ is the reward value obtained.
RL in Traffic Signal Control Problem can be classified according to the following

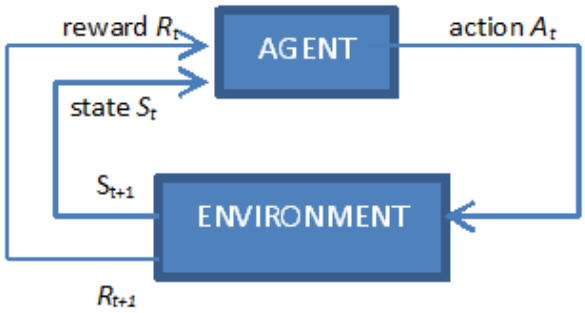

Fig. 1. Reinforcement Learning

parameters:

- Using the traffic flow model : model based, model-free
- By the presence of a central control: centralized, distributed
- By the level at which the calculations are carried out: global and local methods
- Considering network size and architecture: isolated [6] [7][1][8], multi [5][4] and arterial intersections
- Collaborative, not colloborative
- Efficiency criteria: queue [1] and delay based methods [8][6][4].

The rest of the paper is organized as follows. Section 2 gives the problem formulation as a task of RL. Section 3 introduces the proposed Reinforcement Learning Intersection Controller (RLIC) with DQN and explains how it works. Section 4 presents the conducted experiments and results evaluation. Finally, conclusions are given in section 5.

2 Problem Formulation

The aim of the given work is to build an intelligent traffic signal controller for an isolated intersection using RL approach. The model trained using Deep Q-learning (DQN). Traffic signal controller considered as an agent, intersection as an environment. The goal of the agent is to reduce the queue length on the intersection and keep the balance between the all directions. Since the environment is non-stationary traffic controller should be adaptive and be able to adjust to the dynamically changing traffic road situation. DQN is adopted to enlarge state space and suitable to be extended to a transport network of any size.

2.1 State space

The state space element represents number of waiting vehicles in each direction and green phase signal for non-conflicting directions. The links D10 and D30 are considered to have non-conflicting directions, so do D20 and D40 (See Fig. 2). Consequently, at each time step the state signal is represented by a 6-vector of values as follows:

$$NS \doteq D10 \tag{2}$$

$$EW \doteq D20 \tag{3}$$

$$SN \doteq D30 \tag{4}$$

$$WE \doteq D40 \tag{5}$$

$$grPhaseNS \doteq \text{green phase duration for } D10 \text{ and } D30 \tag{6}$$

$$grPhaseWE \doteq \text{green phase duration for } D20 \text{ and } D40 \tag{7}$$

$$S \doteq (NS, SN, WE, EW, grPhaseNS, grPhaseWE) \tag{8}$$

Thus, the state space $\mathcal{S}$ in this setting is unbounded:

$$\mathcal{S} \doteq \{(s_0, s_1, s_2, s_3, s_4, s_5) \mid s_0, s_1, s_2, s_3, s_4, s_5 \in Z^+\} \tag{9}$$

Besides the number of halting vehicles and current phase duration a variety of other factor influencing decision making might be included into the state vector, such as the current, time, weather, etc. This might be a major improvement to the accuracy of decision making and will be considered in the future work.

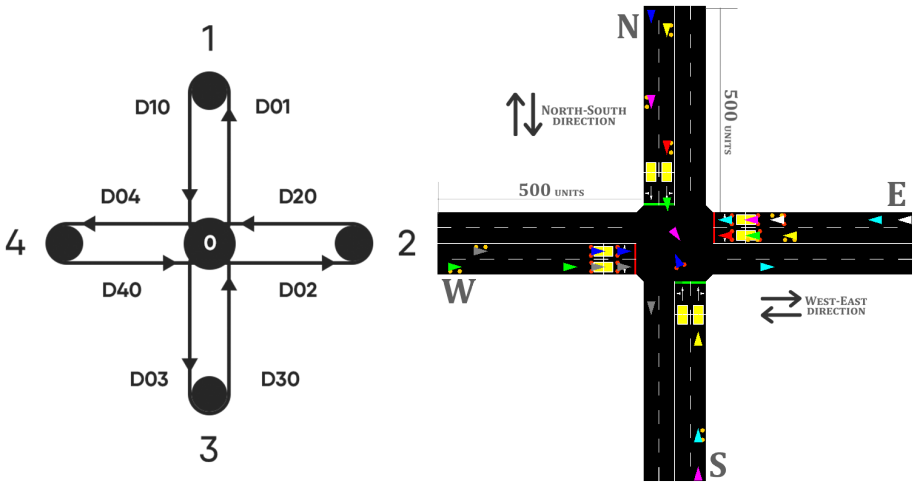

Fig. 2. The test transport network

2.2 Action space

At the beginning of each cycle the agent first receives a state signal, then it performs an action, and, finally, it gets a reward for the action. The agent-environment interaction is an infinite horizon process (which does not naturally break into episodic tasks and continues without limits). The intersection has the green, red and amber phases. The signal plan of each cycle comprises the following components:

1. Green (NS) - Red (WE)
2. Amber
3. Green (WE) - Red (NS)
4. Amber

We modify only the green and red phases, while the amber remains constant.Action element is a precise value of the green phase duration in the non-conflicting directions, therefore, cycle length is non-fixed. The minimum duration is 12 seconds, the maximum is 60. Step size is dt. After each cycle the agent can choose one of the action from the action space:

- $a_0 \doteq (12, 12)$
- $a_1 \doteq (12, 12 + dt)$
- ...
- $a_{n-1} \doteq (60 - dt, 60)$
- $a_n \doteq (60, 60)$

These actions change the signal plan of the next cycle.

$$\mathcal{A} \doteq \{a_0, a_1, a_2.., a_n\} \tag{10}$$

2.3 Reward formula

Insofar as the task of the agent is to reduce the number of halting vehicles as much as possible, while keeping the balance between the conflicting directions, the reward formula ought to clearly reflect these goals. For this task in our previous work [1] we used the following continuous reward formula:

$$S_t = (NS, WE) \rightarrow S_{t+1} = (NS', WE'), \tag{11}$$

the reward signal is defined as follows:

$$\begin{aligned} R(S_t, S_{t+1}) \doteq \beta \overbrace{(|NS - WE| - |NS' - WE'|)}^{\text{equilibrium term}} + \\ +(1-\beta) \underbrace{(NS + WE - (NS' + WE'))}_{\text{queue reduction term}}, \end{aligned} \tag{12}$$

where $\beta \in [0, 1]$ is the trade-off between the queue reduction and equilibrium terms. Throughout all the proposed algorithms we have used $\beta = 0.5$, i.e. the queue reduction and equilibrium goals are equally important for learning.

The reward formula (12) captures the following intuition: if the difference between the conflicting directions has decreased, then the first term will be positive; if the total number of standing vehicles has decreased, then the second term will be positive, and the reward value will also be positive in proportion to the magnitude the mentioned values have decreased. In case the values of both goals increase, the overall reward will be negative in the magnitude of the increases. Finally, when there is an increase in the value of one goal and decrease in the other, the two terms of the formula will eat each other and the resultant value will indicate whose impact is stronger — this accounts for the presence of the plus sign in the formula. However, if we consider a case where the total number of halting vehicles in the NS and SN directions is increasing/decreasing, but distribution is high in one direction and low in other, the proposed reward formula does not take this into account. In addition, if the number of halting vehicles is large and during the next time interval it increases/decreases slightly, and if the number of halting vehicles is small the system can not distinguish these cases. Since, the queue reduction term will give the same results. For solving described problems, we decided to distinguish four directions and propose to use two continuous reward formalas. The first one is based one the queue lenghts:

$$S_t = (NS, SN, WE, EW) \rightarrow S_{t+1} = (NS', SN', WE', EW'), \tag{13}$$

We divided the previously used NS and WE directions into two components NS, SN and WE, EW, respectively.
The reward signal is defined as follows:

$$\begin{aligned} R(S_t, S_{t+1}) \doteq \beta \overbrace{(min(\mu - NS', \mu - SN', \mu - WE', \mu - EW'),)}^{\text{equilibrium term}} + \\ +(1-\beta) \underbrace{(-max(NS', SN', WE', EW'))}_{\text{queue reduction term}}, \end{aligned} \tag{14}$$

where

$$\mu = \frac{(NS + SN + WE + EW)}{4} \tag{15}$$

Minimum from the deviation of the mean in all directions gives information about how much the intersection is in balance.The queue reduction term considers only the current state in order to see the worst state from all directions. Now, the value of our reward in the best case will be 0. When all four directions have the same number of waiting vehicles or there are no ones, first term will be zero. The second term is zero when there is no queue.
The second reward formula based on the waiting time. It is common reward representation used in the previous researches:

$$R(S_t, S_{t+1}) \doteq \sum_{i=1}^{N} \frac{w_t^i}{N_t} - \sum_{i=1}^{N} \frac{w_{t+1}^i}{N_{t+1}} \tag{16}$$

where w is the total waiting time of vehicles in the intersection in current and previous states. N - number of halting vehicles.

3 Proposed method: RLIC with DQN

There are numerous basic approaches to solve the reinforcement learning task. If the environment's dynamics is known, the methods of *dynamic programming (DP)* [2, Chap 4] are employed. If the task is episodic [2, Sec 3.3], the *Monte Carlo* methods are frequently used which involve estimating v_* by averaging sample returns obtained at the end of each successive episode [2, Chap 5]. However, neither of these approaches is suitable to our task, since the model of the environment is not known beforehand, and the task is not episodic.

The *temporal-difference learning (TD)* is a combination of both the dynamic programming and Monte Carlo methods. Like DP, TD methods update estimates of the value functions obtained in turn from other learned estimates. However, in contrast to DP, TD assumes no predefined model of the environment whatsoever. Like Monte Carlo, TD methods can learn directly from raw experience. However, in contrast to Monte Carlo, the TD approach updates the estimates in the *online* fashion, meaning that it does not wait until the end of an episode, but rather does this on each time step (i.e. *bootstraps*).

The TD method estimates the true expected value by averaging the value of $R_{t+1} + \gamma v_\pi(S_{t+1})$ at each time step. However, the value $v_\pi(S_{t+1})$ is not known, so its current estimate $V(S_{t+1})$ is used instead. Upon transition to a new state S_{t+1} and receiving R_{t+1}, the TD methods make the following update:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right] \tag{17}$$

Another perspective on the above formula is:

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t \tag{18}$$

$$\delta_t \doteq \mathit{Target} - \mathit{Estimate} \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{19}$$

Thus, the update shifts the current estimate in the direction of the error between the target at time t and the current estimate of the value function. This error δ_t is called the *TD error*. Notice that the TD error at each time is the error in the estimate made at that time. Because the TD error depends on the next state and the next reward, it is not actually available until one time step later. The update formula (17) follows the following general rule that occurs frequently throughout reinforcement learning:

$$\mathit{NewEstimate} \leftarrow \mathit{OldEstimate} + \mathit{StepSize}\,[\mathit{Target} - \mathit{OldEstimate}] \tag{20}$$

This general update rule can be treated as averaging of the old estimate and the target with the constant parameter $\mathit{StepSize} = \alpha$ that defines the extent to which the new target affects the average.

The majority of TD methods estimate q_* instead of v_*, since we cannot construct a policy derived from the estimate of v_* without the model of the environment. The estimation of q_* is done using essentially the same TD update (17) described above. One of the off-policy TD algorithms is *Q-learning* [2, Sec 6.5]. The target policy for Q-learning is the greedy policy, whereas the behavior policy may be any policy derived from Q which ensures exploration, i.e. all the state-action pairs are continually visited and updated. The update rule for Q-learning appears as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{21}$$

If the state and action spaces are finite, the function Q can be represented as a table. In our solution the estimate of the true action-value function is represented not as a table, but rather as a parametrized continuous function — the neural network. We propose a model-free, online, off-policy reinforcement learning algorithm based on the temporal-difference learning:

$$q_*(s, a) \approx \hat{q}(s, a, w), \tag{22}$$

where $\hat{q}(s, a, w)$ represents the value of the neuron corresponding to the action a in the output of the neural network evaluated at state s. The neural network is parametrized by the set of weight matrices $w \doteq \{w^{(1)}, w^{(2)}, \ldots, w^{(L-1)}\}$, where L is the total number of layers in the neural network. The matrix $w^{(k)}$ for $k = \overline{1, L-1}$ defines the transition from the k^{th} to the $(k+1)^{th}$ layer.

In order to be able to train the neural network, we need to define the cost function. The algorithm should adjust the weight matrices w so as to reduce the cost function as much as possible. The natural way to define a cost function which is consistent with our goal of approximating the optimal action-value function q_* is:

$$J(w) \doteq \frac{1}{2} \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \left(q_*(s, a) - \hat{q}(s, a, w) \right)^2 \tag{23}$$

However, the true values of $q_*(s, a)$ in the sum above are not known, so this cost function is inappropriate. To remedy this state of affairs, the temporal-difference learning comes in.

The algorithm proceeds in the online manner: at each time step t it makes a TD estimate $U_t(S_t, A_t)$ to the target value $q_*(S_t, A_t)$ as follows:

$$U_t(S_t, A_t) \doteq R_{t+1} + \gamma \max_{a' \in \mathcal{A}} \hat{q}(S_{t+1}, a', w_t) \tag{24}$$

Afterwards, we define the cost function as

$$J_t(w_t) \doteq \frac{1}{2} \left(U_t(S_t, A_t) - \hat{q}(S_t, A_t, w_t) \right)^2, \tag{25}$$

which involves the difference between the new and current estimates of the action-value function, with the latter produced by the neural network with the weights at time t. Stochastic gradient descent is used to update the weights of the neural network.

4 Experiments and results

The environment for the experiments is a traffic network in the Figure 1. An isolated intersection build in simulation platform SUMO DLR [3]. Average lenghts of links to 100 meters. We are excluding the U-turns, public transport, pedestrians, traffic rules violations, traffic road accidents, and parking. Travel demand models are given in the Table 3. Initial signal plan: the green phase in NS direction is set to 24 seconds, and the green phase in WE direction is set to 42 seconds. Our simulation runs approximately 28 hours. The parameter settings are given in the Table 1 and Table 2.

Table 1. Parameter setting.

Free parameter	**Value**
Action time interval Δt	6 seconds
Learning rate for Q-learning α_1	0.6
Discount factor for reward γ	0.8
ϵ	0.1

For the evaluation of our method with different rewards using the average queue lenghts and waiting time. Fixed-signal control with predefined phase split is considered as ground truth.

Two group of experiments conducted using synthetic data (Table 3). In the first part of our experiment, uniform distributed demand model (Configuration 1) used as an environment. The result of the simulations is given in the Table 4. The DQN model with proposed rewards and fixed-time controller are compared.

Table 2. The NN parameters.

Parameter	**Value**
Number of layers	4
Number of neurons	6, 14, 34, 81
Learning rate α_2	0.001
Activation function on hidden layers	ReLu
Activation function on output layer	Linear

Table 3. Demand models

Config	**NS**	**SN**	**WE**	**EW**
1 (0-100000 s)	0.2	0.2	0.2	0.2
2 (0-100000s):				
(0-20000 s)	0.2	0.2	0.2	0.2
(20001-40000 s)	0.4	0.4	0.2	0.2
(40001-60000 s)	0.2	0.2	0.2	0.2
(60001-80000 s)	0.2	0.2	0.4	0.4
(80001-100000 s)	0.2	0.2	0.2	0.2

Table 4. Performance on Configuration 1.(QL-queue length, DT-delay time)

Model	**QL NS**	**QL WE**	**QL SN**	**QL EW**	**DT NS**	**DT WE**	**DT SN**	**DT EW**
DQN: Reward 1 (12)	5.64	14.23	5.72	11.53	112.57	369.63	113.68	290.99
DQN: Reward 2 (14)	6.21	8.19	6.25	9.03	135.82	195.65	135.25	223.35
DQN: Reward 3 (16)	3.35	4.56	3.37	4.71	50.26	76.71	50.58	80.19
Fixed-time control	5.37	5.31	5.42	5.24	92.93	92.26	93.50	90.03

For the fixed control several signal plans were tested, and the best one was chosen: (24s, 24s). The Reward 1 is oscilate around zero and decreasing (Fig. 3). During the experiment it was noticed that the signal plan had stopped changing. However, the intersection was overloaded. A situation occured when our reward formula incorreclty reflects the state of the environment. The Reward 2 allows to avoid this problem. The maximum value of this reward may be zero.
Experiments show that the system with the proposed rewards perform near-optimal and able to stabilize when using uniform distributed demand. In the second group of experiments we used mixed demand model, where arrival rate of vehicles is changing during simulation time. The results show that system can not properly arrange to the new situation and reward value has large deviation (Fig. 4). The controller is being rebuilt all the time and cannot stabilize There may be several reasons:

- lacks of NN flexibility.
- often update (occurs each step)
- statistics (history) are not taken into account

Table 5. Performance on Configuration 2. (QL-queue length, DT-delay time)

Model	QL NS	QL WE	QL SN	QL EW	DT NS	DT WE	DT SN	DT EW
DQN: Reward 1 (12)	65.86	25.65	66.16	26.33	1843.41	475.52	1858.93	529.27
DQN: Reward 2 (14)	56.21	34.47	56.85	34.48	1460.05	888.77	1509.09	2932.12
DQN: Reward 3 (16)	55.52	31.41	54.62	31.16	1257.15	717.91	1171.41	714.81
Fixed-time control	53.03	29.18	53.15	29.15	991.66	531.02	985.22	525.54

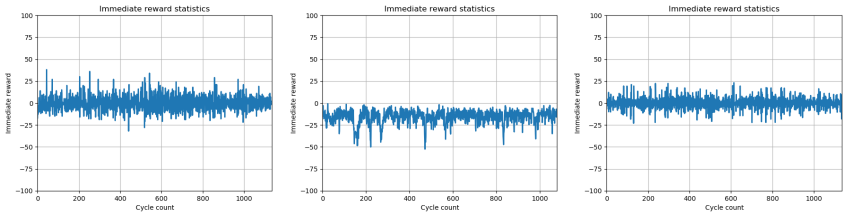

Fig. 3. Configuration 1. Reward 1, Reward 2, Reward 3

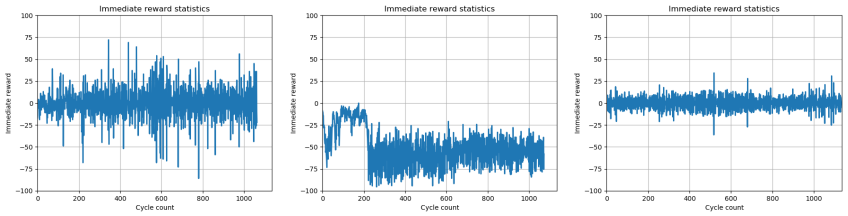

Fig. 4. Configuration 2. Reward 1, Reward 2, Reward 3

5 Conclusion

In this paper, we propose a model-free, online, off-policy reinforcement learning traffic control system. The model is training based on the temporal-difference learning with ANN. The main bias was on the reward function. Based on the shortcomings of the previous work, a new reward formula was proposed. The adaptive controller gave near-optimal performance. The experiments were conducted on a uniform and mixed demand model. The result shows that the system based on the neural network has difficulties in a non-uniform environment. Some reasons were identified. To eliminate the disadvantages, it is planned to further

expand the neural network architecture and consider the possibility of describing the state of the environment with more stable information based on statistics.

Notes and Comments. The work is carried out as a part of research work "Software development for 3D modeling, online monitoring and prediction of the air consumption level in urban and industrial areas" №$BR05236316$, performed at the Satpayev University according to the financial program, designed for 2018-2020 years.

References

1. Kurmankhojayev,D., Tolebi,G., Suleymenov, N.: Online model-free adaptive traffic signal controller for an isolated intersection. In: IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), pp. 109-112. IEEE Press, Novosibirsk Akademgordok (2017).
2. Sutton,R.S.,Barto, A. G.: Reinforcement Learning: An Introduction, The MIT Press, (2016)
3. Krajzewicz, D., Erdmann, J., Behrisch,M., Bieker,L., Recent Development and Applications of SUMO - Simulation of Urban MObility, International Journal On Advances in Systems and Measurements,vol. 5, p.128–138, (2012)
4. Wiering, M.: Multi-agent reinforcement learning for traffic light control, In Proceedings of the 17th ICML, pp. 1151-1158, Morgan Kaufmann, The Netherlands (2000)
5. Chu, T.: Real time Reinforcement Learning in Traffic Signal System, https://pdfs.semanticscholar.org/3004/bda51db2c58a89c31aa8c908520aaf7ee29d.pdf
6. Abdoos, M., Mozayani, N., Bazzan, Ana.L.C.: Traffic Light Control in Non-stationary Environments based on Multi Agent Q-learning, In: 14th International IEEE Conference on Intelligent Transportation Systems, pp.1580-1585, Washington, (2011)
7. Yaping, W., Zheng, Zh.: A method of Reinforcement Learning Based Automatic Traffic Signal Control, In: Third International Conference on Measuring Technology and Mechatronics Automation, pp.119-122, IEEE Press, (2011) . DOI10.1109/ICMTMA.2011.35
8. Araghi, S., Khosravi, A., Johnstone, M., Chreighton, D.: Intelligent Traffic Light Control of Isolated Intersections using Machine Learning Methods , In: IEEE International Conference on Systems, Man, and Cybernetics, pp.3621-3622, IEEE Press, (2013) . DOI310.1109/SMC.2013.617