# A PROPOSAL OF WEIGHTED Q-LEARNING FOR CONTINUOUS STATE AND ACTION SPACES

**Yuhu Cheng, Jianqiang Yi, Dongbin Zhao**

*(Laboratory of Complex Systems and Intelligence Science,*
*Institute of Automation,*
*Chinese Academy of Sciences, Beijing 100080, China)*

Abstract: A kind of weighted Q-Learning algorithm suitable for control systems with continuous state and action spaces was proposed. The hidden layer of RBF network was designed dynamically by virtue of the proposed modified growing neural gas algorithm so as to realize the adaptive understanding of the continuous state space. Based on the standard Q-Learning implemented by RBF network, the weighted Q-Learning was used to solve the control problem with continuous action outputs. Simulation result of mountain car control verified the validity of the proposed weighted Q-Learning algorithm.


Keywords: continuous state space, continuous action space, weighted Q-Learning, neural gas algorithm, RBF network, mountain car.

## 1. INTRODUCTION

Reinforcement learning is a kind of learning algorithm between supervised and unsupervised learning algorithms which is based on markov decision process (MDP). For the solution of large-scale MDPs or continuous state and action spaces, it's impossible for reinforcement learning agent to go through all the states and actions. In order to realize the optimal approximation for value functions of continuous states and actions respectively, therefore, learning agent must have generalization ability. In other words, such an agent should be able to utilize finite learning experience to acquire and express a good knowledge of a large-scale space effectively, see (Sutton, *et al*., 1999). How to design a function approximator with abilities of high generalization and computation efficiency has become a key problem for the research field of reinforcement learning.

Q-Learning is an effective reinforcement learning method to solve markov decision problem with incomplete information. It is also viewed as a kind of asynchronous dynamic planning method. Since Watkins and Dayan (1992) proposed Q-Learning algorithm and proved its convergence, it has been received broad attention. However, research works about Q-Learning are restricted to limited markov decision problem and there are little works related to continuous state and action spaces. Because states and actions of many stochastic control systems are continuous in fact, we should make state and action spaces discrete when we use Q-Learning to solve the stochastic optimal control problem of continuous state and action spaces. But such a discrete operation is likely to result in the following problems:(a) hidden state problem easily occurs when state space is divided roughly;(b) curse of dimensionality problem appears when the state space is enormous;(c) the property of markov of the system can't be guaranteed after discrete operation (Munos, 1997). The surrounding environment and the task are very complex; therefore, we can't obtain satisfactory control effects by merely defining a simple discrete action set. Moreover, if we make continuous action space discretization for an actual control system even disregarding spatial and computational complexities, the pre-defined discrete action set may not includes the optimal action of each time-step so as to satisfactory control effects are difficult to obtain.

Lately, some scholars pursued researches on extension of the application field of Q-Learning to continuous state and action spaces. The main difference between state and action spaces is that the input data used to constitute state space is provided by environment, but on the other hand action space is obtained only by virtue of exploring by learning agent. Werbos (1992) proposed a kind of adaptive critic method by using several feedforward networks to realize Q-Learning. Each element of action vector is assigned reinforcement signal through learning system dynamics. The method doesn't satisfy model-free condition; therefore, if the system dynamics model is known, such a model-based adaptive critic method is an effective reinforcement learning method suitable for continuous state and action spaces. Santamaria *et al.* (1998) put forward a kind of reinforcement learning method based on CMAC. Because CMAC has stronger nonlinear approximation and generalization abilities, it can realize generalization of reinforcement learning in continuous state space. The inputs of CMAC are states and actions, and its outputs are desired values. The method has the following defects. At first, we should explore all possible action spaces in order to find out the action with the maximal Q value, therefore the method couldn't meet online learning requirement. Moreover, the number of weights of CMAC network increases exponentially according to the dimension of state because the learning algorithm based on CMAC is a kind of overlap state discrete algorithm. At last, the convergence of direct gradient descent reinforcement learning algorithm hasn't been proven from the point of view of theory. Claude (1997) proposed a Q-Learning system based on Kohonen network where states, actions and desired values are all viewed as eigenvectors. Action selection is determined by selecting the optimal matching units, but the action is merely piecewise continuous and in fact doesn't completely satisfy continuous standard. Smith (2002) used two SOM to approximate state and action spaces respectively, and utilized one-step TD error of Q-Learning to adjust exploring orientation, the center of units of actions and inputs mapping. The algorithm is quite complex due to coordinator problem between the two SOM networks. Santos (1999) put forward a kind of reinforcement learning algorithm based on radial basis function (RBF) network. Each RBF unit has a central vector that is similar to eigenvector of Kohonen network. The number of hidden units of RBF network is equal to the number of actions that are possiblely adopted by learning agent. The action is piecewise continuous. Samejima and Omori (1999) gave a state space adaptive division method called Actor-Critic method. The method can adaptively increase the number of units of the hidden layer according to the surrounding environment and task. At the beginning of learning, the hidden layer only has one unit and one basis function. During the learning process, new basis functions will be produced by automatically dividing state space according to TD error till the requirements of the task are met. A Q-Learning system based on dynamic neural field was proposed by Gross *et al.* (1998). This Q-Learning system uses neural gas to cluster similar states, and uses neural field to code action values and then selects the action with the maximal Q value. Despite the method meets state and action generalization standards, low velocity of action selection hampers its application.

Aiming at effective control of systems with continuous state and action spaces, a weighted Q-Learning is proposed in the paper. We use RBF network to approximate the utility values of discrete actions, and then obtain continuous actions that actually act upon the system by making each discrete action weighted according to its utility value. Because the structure and parameters of RBF network are vital to approximation effects, we propose a modified growing neural gas (GNG) algorithm to realize adaptive design of the hidden layer of RBF network. At last, simulation research of mountain car control is given and the simulation result verifies that the proposed weighted Q-Learning algorithm has better learning efficiency and generalization capability.

## 2. WEIGHTED Q-LEARNING BASED ON RBF NETWORK

The idea of Q-Learning is to directly optimize a Q function that can be computed iteratively and not to approximate environment model directly. At each time step, the agent observes the state $s_t$, takes the action $a_t$ subsequently, it then receives a reward on the new state $s_{t+1}$. The reward is discounted into the future, meaning the rewards received $n$ time steps later are worthless by $\gamma^n$ than that receives at present, $\gamma \in (0,1)$ is discounting factor. Thus the cumulative discount reward is given by: During the exploration, agent updates its Q-values based on the received reward $r(s_t, a_t)$ and the perceived state transition from $s_t$ to $s_{t+1}$ using:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[ r(s_t, a_t) + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{1}$$

where $\alpha \in (0,1)$ is the learning rate that specifies the incremental update.

There are many realization forms of Q-Learning by virtue of neural network. We can set codes of states and actions as network inputs, and then apply a definite learning rule to train the network to produce the goal value of Q. We also can train an individual network for each action, and set states and Q value as inputs and output respectively. In addition, there is a commonly used method to train a network whose inputs and outputs are states and Q values corresponding to each action. The key problem of realizing Q-Learning by neural network is learning algorithm. Only the premise of the optimal strategy is achieved, can the definition of Q function, i.e. Eq.(1)

is established. Eq.(1) isn't established during the learning process. Here, we define TD error as $\delta_t$.

$$\delta_t = \Delta Q = r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \quad (2)$$

where $Q(s_{t+1}, a)$ stands Q value corresponding to the next state. $\delta_t$ reflects the degrees of good or bad of the selected action. $\delta_t$ can be set as small as possible by adjusting network weights.

The proposed system structure is a five-layer network as shown in Fig.1. The signal propagation and the basic function of each layer are described as follows.

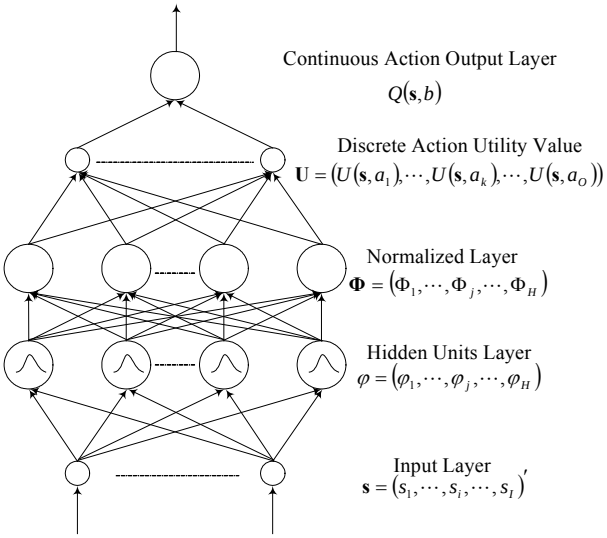


Fig. 1. System structure

Layer 1: input layer. $s_t = (s_1, \cdots, s_i, \cdots, s_I)^t \in R^I$ denotes input state vector at time $t$ and $I$ denotes the dimension of the input vector. The number of input units is equal to the dimension of the input vector.

Layer 2: hidden units layer that is composed of $H$ radial basis function units. The hidden layer is constructed adaptively based on a modified GNG algorithm proposed in this paper. The number of hidden units isn't predefined, but is adjusted dynamically by the modified GNG algorithm during the learning process. The active function of hidden unit is Gaussian function, therefore the output of the j[th] hidden unit is

$$\varphi_j(s_t) = \exp\left[ -\frac{\left\| s_t - \mu_j \right\|^2}{2\sigma_j^2} \right], \quad j = 1,2,\cdots,H \quad (3)$$

where $\mu_j$ and $\sigma_j$ are the center (mean) and the width (variance) of the j[th] hidden unit respectively, and $H$ denotes the number of hidden units.

Layer 3: normalized layer that is composed of $H$ units. The normalized output corresponding to the j[th] hidden unit is

$$\Phi_j(s_t) = \frac{\varphi_j(s_t)}{\sum_{l=1}^{H} \varphi_l(s_t)}, \quad j = 1,2,\cdots,H \quad (4)$$

Layer 4: linear approximation layer that is composed of $O$ units. This layer fulfills approximation task for $O$ discrete action utility values $U(s_t, a_k)$, i.e. Q values of actions. This approximation represents contributes of each discrete action to the compounded continuous action.

$$U(s_t, a_k) = \sum_{j=1}^{H} w_{jk} \Phi_j(s_t), \quad k = 1,2,\cdots,O \quad (5)$$

where $w_{jk}$ is connect weight from the j[th] output of the normalized layer to the k[th] unit of the linear approximation layer.

Layer 5: continuous action output layer that is composed of only one unit (can be extended to the application of continuous action vector output). The output of continuous action depends on discrete actions. Differing from winner-take-all mechanism and $\varepsilon-$ greedy strategy, command fusion mechanism is used to select continuous actions. Command fusion mechanism weights all discrete actions according to their utility values to form a continuous control action that represents the consensus of the viable action set. Thus, this approach provides for a coordination scheme that allows all the behaviors to simultaneously contribute to the control of the system in a cooperative rather than a competitive manner. If we set utility value of each discrete action at state $s_t$ as $U(s_t, a_k)$, then action $b_t$ that is acted upon the system at time $t$ takes the value spanned in the range of $\left[ \min_{i \in O} a_i, \max_{j \in O} a_j \right]$. The computation equations of $b_t$ and its Q value are

$$b_t = \frac{\sum_{k=1}^{O} a_k u_k}{\sum_{l=1}^{O} u_l} \quad (6)$$

$$u_k = \frac{1}{1 + \left( \max_{g \in O} U(s_t, a_g) - U(s_t, a_k) \right)^2} \quad (7)$$

$$Q(s_t, b_t) = \sum_{k=1}^{O} U(s_t, a_k) u_k \quad (8)$$

where $k \in [1, O]$ and $O$ is the number of discrete actions, and $u_k$ is the weight coefficient of action $a_k$. We can see from Eq. (8) that the Q value of continuous action is obtained by combining the utility values of all the discrete actions using linear combinations.

TD error is defined as

$$\delta_t = r_t + \gamma Q(s_{t+1}, b_{t+1}) - Q(s_t, b_t) \quad (9)$$

where $r_t$ is the future reinforcement signal when action $b_t$ is executed at state $s_t$. Furthermore, system state will transit from state $s_t$ to state $s_{t+1}$.

The square sum of the TD errors $E_t = \frac{1}{2}\delta_t^{\,2}$ is used as the basis of weights updating to reduce TD error. The weights updating rule is

$$w_{jk}(t+1) = w_{jk}(t) - \eta \frac{\partial \mathrm{E}_t}{\partial w_{jk}}$$
$$= w_{jk}(t) + \eta\delta_t\left(\frac{\Phi_j(s_t)}{u_k} + 2\Phi_j(s_t)U(s_t,a_k)\sqrt{\frac{1-u_k}{u_k}}\right) \tag{10}$$

where $\eta$ is learning rate.

## 3. STATE SPACE GENERALIZATION BASED ON MODIFIED GNG ALGORITHM

The structure and parameters of RBF network are vital to the system approximation effects. We propose a modified GNG algorithm based on GNG algorithm that was first put forward by Friztke (1995) .The modified GNG algorithm is used to construct the hidden layer of RBF network. In order to obtain appropriate approximation precision, we don't pre-set the number of hidden units of RBF network, but use the modified GNG algorithm to make the network adaptively add or delete hidden units with the requirements of the environment and task so as to realize adaptive, online state space division in deed. This approach of construction of hidden layer of RBF network using the modified GNG algorithm has self-organized and self-learning abilities whose structure is shown in Fig.2.

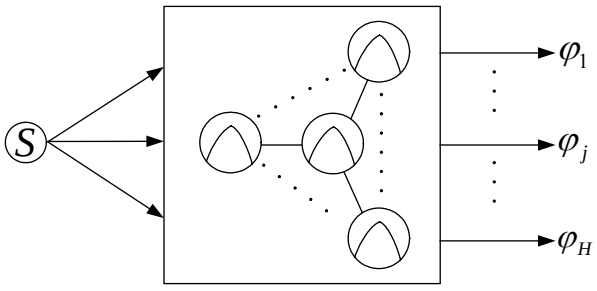


Fig. 2. Structure of construction of hidden layer of RBF network by modified GNG algorithm

The modified GNG algorithm is described as follows.

Each hidden unit has the following parameters: reference vector (denoting the position of input unit that is corresponding to hidden unit on input space, i.e. weights), edge (denoting the neighbor of the topology structure on space which is used to connect hidden units), age (denoting the existence term of edge), local error and local square error (determining new unit addition, and here using TD error to define the local error and local square error of units).

1. Initializing GNG algorithm. Establishing two hidden units corresponding to stochastic positions and connecting the two units using edge. The age of the edge is set as 0, and the local error and local square error of units are also initialized to 0.
2. Submitting input vector $s_t$ and computing the Euclidean distances $\|s_t - v_i\|$ between $s_t$ and each unit. Determining the optimal matched unit $m$ and the sub-optimal matched unit $n$ according to the least Euclidean distance rule. Supposing that the reference vectors corresponding to units $m$ and $n$ are $v_m$ and $v_n$ respectively, then we can get
$$m = \arg\min_{m\in H}\|s_t - v_m\|, n = \arg\min_{m\in H, n\neq m}\|s_t - v_n\| \tag{11}$$
3. Defining local error $f_m$ and local square error $g_m$ of the optimal matched unit to evaluate the adaptability of the unit. The updating equations are
$$f_m(t+1) = (1-\beta_1\Phi_m(s_t))f_m(t) + \beta_1\Phi_m(s_t)\varphi_m(s_t)\delta_t \tag{12}$$
$$g_m(t+1) = (1-\beta_2\Phi_m(s_t))g_m(t) + \beta_2\Phi_m(s_t)\varphi_m(s_t)\delta_t^2 \tag{13}$$
where $\beta_1$ and $\beta_2$ are learning rates of $f_m$ and $g_m$ respectively.
4. Moving the optimal matched unit $m$ and its neighbour units (all units that are connected with unit $m$ by edges) along the direction $s_t$ so as to adjust the center of hidden unit of RBF network.
$$v_m \leftarrow v_m + \tau_m\delta_t(s_t - v_m), v_i \leftarrow v_i + \tau_i\delta_t(s_t - v_i)$$
$$\forall i \in Neighbour(m). \tag{14}$$
where $\tau_m, \tau_i \in [0,1]$ and $\tau_m >> \tau_i$.
5. Adjusting the widths of the unit $m$ (after being moved) and its neighbour units. Defining the width of the j[th] hidden unit as the average value of all the distances between unit $j$ to its all neighbor units, that is
$$\sigma_j = \frac{1}{N_j}\sum_{j=1}^{N_j}\|v_j - v_c\|, \quad \forall c \in Neighbour(j) \tag{15}$$
where $N_j$ denotes the number of the neighbour units of unit $j$.
6. Increasing the age of edges that connect unit $m$ with its neighbour units. If units $m$ and $n$ are connected by edge, the age of the edge is set as 0. On the other hand, if units $m$ and $n$ aren't connected, connect the two units and set the age of the edge is 0. Any edge whose age is bigger than $age_{\max}$ should be deleted. If there is a unit

having no edges to be connected, the unit is viewed as a 'dead' unit and should be deleted. The widths of those units influenced should be re-computed according to Eq. (15).

7. Adding a new unit when the activation degree, $\varphi_m$ of the optimal matched unit is smaller than threshold $\varphi_T$ and $g_m / f_m$ is bigger than threshold $\theta_T$ which indicate that the existing basis functions didn't cover the current input vector. The detailed operation steps are: finding out a unit $u$ with the largest local error; finding out a unit $v$ with the largest local error from the neighbor units of unit $u$; adding a new unit $r$ between units $u$ and $v$, and setting its reference vector is $v_r = \frac{1}{2}(v_u + v_v)$; connecting units $u$ with $r$, $r$ with $v$ respectively; deleting the connect between units $u$ and $v$, and re-computing the widths of the RBF basis function of units $u$, $v$ and $r$.
8. Repeating the above steps in case the stop condition isn't satisfied.

It is noted that the proposed algorithm in the paper is different from the growing algorithm proposed by Fritzke. Fritzke's algorithm works in an offline manner, whereas the modified GNG algorithm is an online learning algorithm. In addition, Fritzke's algorithm adds new units according to fixed iterative step, whereas the modified GNG algorithm adds or deletes new units according to the approximation requirements so as to always make the network structure keep the optimal status.

## 4. SIMULATION RESEARCH

In order to verify the validity of the proposed weighted Q-Learning, a mountain car control with continuous state and action spaces is simulated. Mountain car is usually viewed as a typical control object with continuous state and action spaces so as to verify the learning efficiency and generalization ability of reinforcement learning algorithm.

Mountain car control problem has two-dimension continuous state space and one-dimension continuous action spaces. Supposing that there is no aprior knowledge about the system dynamics besides the state observation values, therefore, the traditional model-based optimal control strategies can't solve the problem. Fig.3 gives the sketch map of mountain car control where the curve denotes the terrain of the valley, and 'Goal' is the goal point. The task of the system is to move the car from any point to 'Goal' point as quick as possible under the condition that the power energy of the car is insufficient. The state variables of the system are the horizontal position $p$ and the horizontal velocity $v$ of the car. The state space meets $\{(p,v) \in R^2 | -1.2 \le p \le 0.5, -0.07 \le v \le 0.07\}$. Control variable is horizontal force acted upon the car, and the action space meets $u \in [-1,1]$. The system dynamics model is adopted as the following equation during simulation.

$$\begin{cases} \dot{p} = v \\ \dot{v} = 0.001u - 0.0025\cos 3p \end{cases} \tag{16}$$

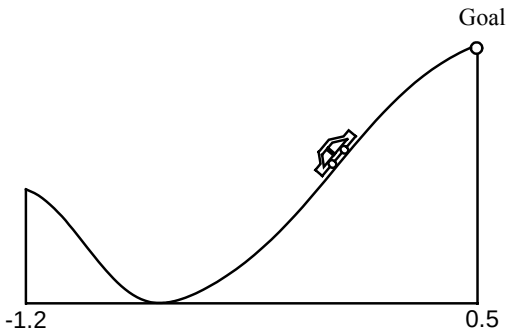


Fig. 3. Sketch map of mountain car control

The goal of the learning controller is to realize the least time control of moving the car from any point to 'Goal' point on the premise of having no aprior knowledge about the system dynamics. The above learning control problem can be modeled by a deterministic MDP, and the rewards function is designed as

$$r_t = \begin{cases} -1, & p < 0.5 \\ 0, & p \ge 0.5 \end{cases}$$

The initial state of the car can be taken from the range $p \in [-1.2, 0.5] \cap v \in [-0.07, 0.07]$ during each trial. The current learning process will end when the car reaches 'Goal' point or time step exceeds the given value. The performance index of the learning system is defined as the time step of moving the car from any point to 'Goal' point. Figures 4, 5, 6, 7 give simulation curves. Fig.4 is position curve of the car. We can see from Fig.5 (velocity curve of the car) that the car can gain the needed potential energy by reverse movement under the condition that the power energy is insufficient. Fig.6 is learning curve indicating the learning steps of moving the car from any point to 'Goal' point. The horizontal coordinate of Fig.6 is learning trials. Seeing from Fig.6, the car can reach the goal point within 200 steps after 60 trials. Fig.7 gives Q value curve after 80 trials. Therefore, it's concluded that the proposed algorithm has better learning efficiency and generalization ability.

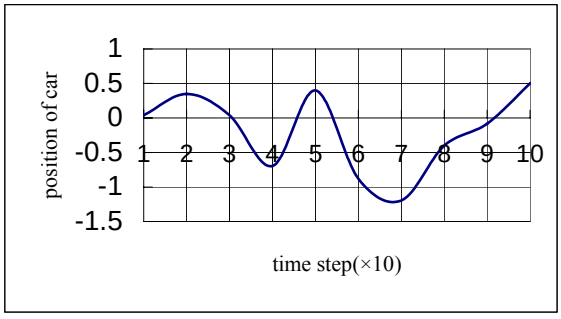


Fig. 4. Position curve of car

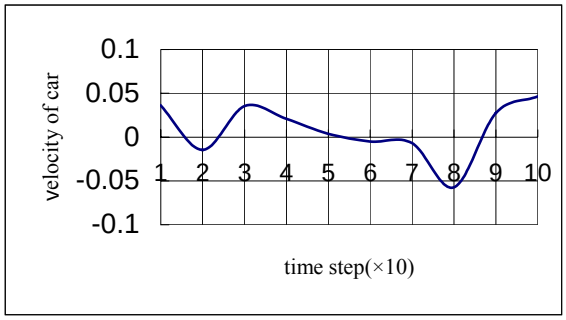


Fig. 5. Velocity curve of car

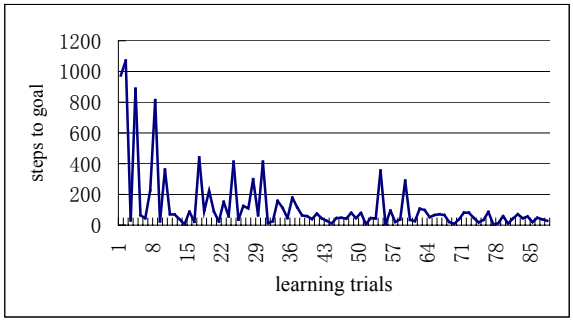


Fig. 6. Learning curve

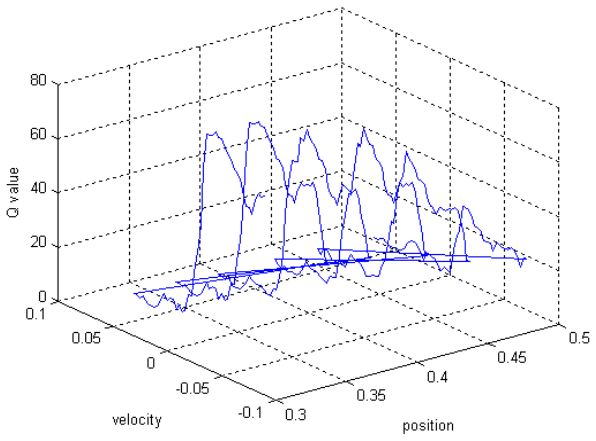


Fig. 7. Q-Value curve

## 5. CONCLUSIONS

A kind of weighted Q-Learning algorithm for continuous state and action spaces was proposed in the paper. RBF network is used to implement standard Q-Learning so as to approximate the utility values of discrete actions. And then, the continuous action acted upon the actual system can be obtained by weighing the utility values of discrete actions according to the proposed weighted rule in the paper. In this way, the application field of Q-Learning is extended to control system with continuous state and action spaces. Regarding the importance of the structure and parameters of RBF network on approximation effects on discrete action utility values, we put forward a modified GNG algorithm and applied it to construct the hidden layer of RBF network. Therefore, the number of hidden units, centers and widths of RBF network can be adjusted adaptively. Simulation result of mountain car control verified the validity of the proposed weighted Q-Learning algorithm.

## ACKNOWLEDGEMENTS

This work was partly supported by 973 Project (Grant No. 2003CB517106) and International Cooperation Key Project (Grant No. 2004DFB02100) of Ministry of Science and Technology, China.

## REFERENCES

Claude, F. T. (1997). Neural reinforcement learning for behavior synthesis. *Robotics and Autonomous Systems*, **Vol. 22, No. 3**, 251-281.

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems 7*, MA: MIT Press, 625-632.

Gross, H. M., V.Stephan, and M.Krabbes. (1998). A neural field approach to topological reinforcement learning in continuous action spaces. *Proceedings of IEEE World Congress on Computational Intelligence,* **Vol. 2**, 826-832.

Munos, R. (1997). A convergent reinforcement learning algorithm in the continuous case based on a finite difference method. *Proceedings of the International Joint Conference on Artificial Intelligence*, **Vol. 1**, 268-278

Samejima, K, and T. Omori. (1999). Adaptive Internal State Space Construction Method for Reinforcement Learning of a Real-world Agent. *Neural Networks*, **No. 12**, 1143-1155.

Santamaria, J. C., R. S.Sutton, and Ashwin Ram (1998). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, **Vol. 6, No.2**, 163—218.

Santos, J. M. (1999). Contribution to the study and design of reinforcement functions. PhD thesis, Universidad de Buenos Aires, Universities d'Aix-Marseille III.

Smith, A. J. (2002). Applications of the Self-organizing Map to Reinforcement Learning. *Neural Network*, **No. 15**, 1107-1124.

Sutton, R.S., D.McAllester, and S.Singh, et al. (2000). Policy Gradient Methods for Reinforcement learning with Function Approximation. *Advances in Neural Information Processing Systems 12* (Proceedings of the 1999 conference), pp. 1057-1063. MIT Press.

Watkins, C.J.C.H. and P. Dayan. (1992). Q Learning. *Machine Learning*, **Vol. 8, No. 3,** 279-292.

Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. (D. A. White and D. A. Sofge, editors).Van Nostrand Reinhold.