2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Simulated Annealing Q-Learning Algorithm for ABR Traffic Control of ATM Networks

Xin Li, Yucheng Zhou, Georgi M. Dimirovski, *Senior Member*, *IEEE*, and Yuanwei Jing

Abstract—**One of the fundamental issues in asynchronous transfer mode (ATM) networks is the congestion problem of information flow. Due to the complexity and variability of ATM, it is difficult to accurately describe the characteristics of source traffic. This paper presents a traffic controller to solving the congestion problem by using Q-learning conjunction with simulated annealing. In stead of relying on the mathematical model for source traffic, the controller is designed to learn an optimal policy by directly interacting with the unknown environment. The simulated annealing is a powerful way to solve hard combinatorial optimization problems, which is used to adjust the balance between exploration and exploitation in learning process. The proposed controller forces the queue size in multiplexer buffer to the desired value by adjusting the source transmission rate of the available bit rate (ABR) service. Simulation results show that the proposed method can promote the performance of the networks and avoid the occurrence of congestion effectively.**

I. Introduction

ASYNCHRONOUS transfer mode (ATM) is a key technology for broadband integrated services digital networks. In order to satisfy various classes of multimedia traffic with different quality of service (QoS) requirements, efficient statistical multiplexing and cell switching, ATM networks must support different service categories [1], namely constant bit rate (CBR) and variable bit rate (VBR) service for real-time applications, unspecified bit rate (UBR) and available bit rate (ABR) service for nonreal-time applications.

However, if many sources send traffic at the same time, the total traffic at the switch may exceed the output capacity causing delays, buffer overflows and cell loss. The congestion of information is the main reason that affects the QoS in ATM networks. Of all the service categories above, ABR is the only one that can be controlled by using the feedback mechanism. ABR service allows applications to fully utilize the available bandwidth in network by adjusting their instantaneous transmission rates to the available capacity. A traffic control scheme is essential for the support of ABR traffic to utilize the available network bandwidth without causing congestion.

The traffic control of ABR service is difficult owing to the uncertainties and highly time-varying of different traffic patterns. The traffic control mainly checks the availability of bandwidth and the buffer space necessary to guarantee the requested QoS. A major problem here is lack of information related to the characteristics of source traffic. Devising a mathematical model for source traffic is the fundamental issue. However, it has been revealed that this is a very difficult task.

In order to overcome the above-mentioned difficulties, some traffic control schemes with learning capability have been employed in ATM networks [2-6]. Its basic advantage is the ability to learn the source traffic characteristics from sufficiently big and representative data samples. But it is obvious that the accurate data, needed to train the parameters, are hard to get for the disturbance and the error in instrument measuring. In this circumstance, the reinforcement learning with the ability of self-learning shows its particular superiority in ABR traffic control [7,8].

In this paper, an ABR traffic controller is proposed to solving the congestion problems in ATM networks. The optimal policy of source transmission rate is obtained through a form of model-free reinforcement learning, known as Q-learning. Instead of relying on a known teacher providing a correct output in response to an input, the controller is designed to learn an optimal policy by directly interacting with the environment [9]. Learning is accomplished progressively by appropriately utilizing the past experience which is obtained during real-time operation. But in the learning process, the balance between exploration and exploitation is one of the key problems of action selection. The simulated annealing has been shown to be an effective approximate algorithm to solve combinational optimization problems. It is introduced into the control of balance between exploration and exploitation [10].

The proposed controller forces the queue size in bottleneck node to the desired value by adjusting the source transmission rate of ABR service. The results of simulation show that the proposed method can promote the performance of networks and avoid the occurrence of congestion effectively.

II. Theoretical Framework

A. Architecture of Traffic Controller Proposed

This section gives the detailed architecture of the proposed

Manuscript received September 20, 2007. This work is supported by the National Natural Science Foundation of China under Grant 60274009 and Specialized Research Fund for the Doctoral Program of Higher Education under Grant20020145007.

Xin Li and Yuanwei Jing are with Faculty of Information Science and Engineering, Northeastern University, Shenyang, Liaoning, 110004, P.R. of China (e-mail: lixin820106@126.com).

Yucheng Zhou is with Department of Research Institute of Wood Industry, Chinese Academy of Forestry, Beijing, 100091, P.R. of China (e-mail: zhouyc@caf.ac.cn).

Georgi M. Dimirovski is with Faculty of Engineering, Computer Engg. Dept, Dogus University of Istanbul, TR-347222 Istanbul, Rep. of Turkey (e-mail: gdimirovski@dogus.edu.tr).

ABR traffic controller as shown in Fig.1.

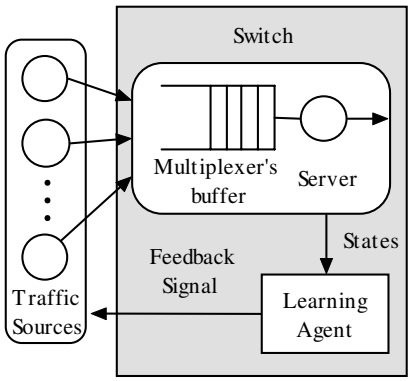

Fig. 1. Architecture of the proposed traffic controller

In ABR traffic control, the traffic controller proposed is called learning agent, and ATM networks make up the environment with which the learning agent works. The inputs of the proposed controller are state variables (S) composed of queue length (q) and source transmission rate (u). The output is the feedback signal (a) to the traffic sources, which means the probability mass function of source rate. The controlled sending rate u_t at sample time t is defined by $u_t = a_t \cdot u_{\max}$, where $u_{\max}$ is the maximum sending rate of traffic sources. Reinforcement signal r for a specified state is denoted as r=1 for reward, and r=0 for penalty.

The learning agent interacts repeatedly with the network environment. During each interaction, the learning agent first observes the states of environment. The learning agent then decides to execute an action on which rate the traffic sources should take. This results in a reward r that is received by the agent and in a transition of the state of network environment from the old state to the new state. The learning agent's reward and the new state of the network environment only depend on the old state and on the action that was executed by the learning agent. In the traffic controller proposed learning agent has no prior knowledge of its environment model. The goal of learning agent is to find an optimal policy for choosing actions.

B. Reinforcement Signal

Q-learning is to learn what to do and how to map situations to actions, so as to maximize a numerical reward signal, which is also called reinforcement signal. So, it is vital to choose an appropriate signal as the reinforcement signal in Q-learning.

In ATM networks, the reward r is evaluated by fuzzy logic rules according to the states of network environment. The real value of reward r, varied as a graded function of system performance, can facilitate the learning speed [11].

The fuzzy reward evaluator (FRE) evaluates the reward for environmental states. FRE relies on three parameters: the current queue length (q), the current rate of queue length change ($\dot{q}$), and the current rate of source transmission rate change ($\dot{u}$) to generate a reward or a punishment for the action in a state. If the system is enhanced toward positive evolution, the action will be rewarded; otherwise punished. The functional blocks of FRE are a fuzzifier, a defuzzifier, and an inference engine containing a fuzzy rule base. The fuzzifier performs the function of fuzzification that translates the value of each input linguistic variable $s(q,\dot{q},\dot{u})$ into fuzzy linguistic terms. These fuzzy linguistic terms are defined in a term set $f(s)$ and are characterized by a set of membership function $\mu(s)$. The defuzzier describes an output linguistic variable of reward (penalty) y_r by a term set $f(y_r)$, characterized by a set of membership functions $\mu(y_r)$, and adopts a defuzzification strategy to convert the linguistic terms of $f(y_r)$ into a nonfuzzy value representing decision reward (penalty) r.

The term set should be determined at an approximate level of granularity to describe the values of linguistic variables. For queue length, the term set is defined as $f(q)$={Low(L), Medium(M), High(H)}, which is used to describe the degree of queue lengths. The term set for the rate of queue length change is defined as $f(\dot{q})$={Decrease(D), Increase(I)}, which describes the rate of queue length change as "Decrease" or "Increase". The term set for the rate of source transmission rate change is defined as $f(\dot{u})$={Negative(N), Positive(P)}, which describes the rate of source transmission rate change as "Negative" or "Positive". On the other hand, in order to provide a precise graded reward in various states, the term for the reward is defined as $f(y_r)$={Penalty More(PM), Penalty Slightly(PS), No Reward(NR), Reward Slightly(RS), Reward More(RM)}. The membership functions (MFs) for terms in the term set are defined with a triangular MF or a trapezoidal MF because their simple formulas and computational efficiency are suitable for real-time operation. The MFs of the term set are shown in Fig.2.

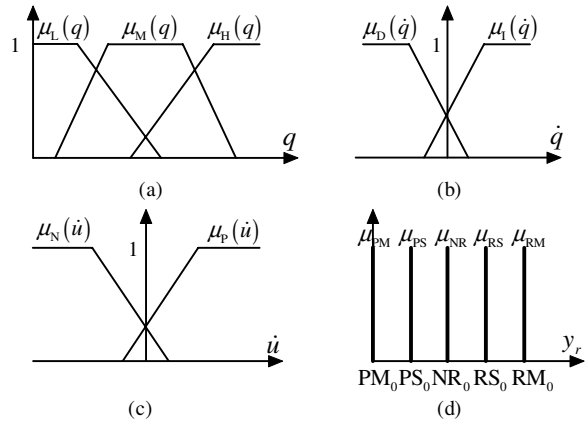

Fig. 2. MFs of the term set (a) $f(q)$, (b) $f(\dot{q})$, (c) $f(\dot{u})$, and (d) $f(y_r)$

The fuzzy rule base is a reward knowledge base, characterized by a set of linguistic statements in the form of "if-then" rules that describe the fuzzy logic relationship between the input variables and the reward (penalty) y_r. According to fuzzy set theory, the fuzzy rule base forms a fuzzy set with dimensions 3×2×2=12. Table I shows a total of twelve inference rules in the fuzzy rule base under various system states. For example, rule 1 can be linguistically started as "if the queue length is low, the queue length change rate is decreased, and the transmission change rate is negative, then give more penalty."

TABLE I
RULE TABLE OF FRE

Rule	q	$\dot{q}$	$\dot{u}$	y_r	Rule	q	$\dot{q}$	$\dot{u}$	y_r
1	L	D	N	PM	7	M	I	N	RS
2	L	D	P	PS	8	M	I	P	RM
3	L	I	N	NR	9	H	D	N	PS
4	L	I	P	NR	10	H	D	P	PM
5	M	D	N	RS	11	H	I	N	PS
6	M	D	P	PM	12	H	I	P	PM

The proposed FRE adopts the max-min inference method for the inference engine because it is designed for real-time operation. An example of the max-min inference method is elaborated in the following. Rules 1, 6, 10, and 12 have the same reward y_r that is "PM". We assume that the current system states are q_0, $\dot{q}_0$, and $\dot{u}_0$. Applying the min operator, we can obtain the membership value of reward y_r=PM of rule 1, denoted by ω_1, as

$$\omega_1 = \min\left[\mu_{\mathrm{L}}(q_0), \mu_{\mathrm{D}}(\dot{q}_0), \mu_{\mathrm{N}}(\dot{u}_0)\right] \cdot \mu_{\mathrm{PM}}(y_r = \mathrm{PM}_0) \quad (1)$$

where PM_0 is the center of the MF $\mu_{\mathrm{PM}}(y_r)$. The membership values of rules 6, 10, and 12, denoted by ω_6, ω_{10}, and ω_{12}, respectively, can be derived in the same manner. Subsequently, applying the max operator yields the overall membership value of reward y_r=PM, denoted by ω_{PM}, as follows:

$$\omega_{\mathrm{PM}} = \max(\omega_1, \omega_6, \omega_{10}, \omega_{12}) . \quad (2)$$

The overall membership values of rewards PS, NR, RS, and RM, denoted by ω_{PS}, ω_{NR}, ω_{RS}, and ω_{RM}, respectively, can be calculated in the same way.

The FRE utilizes the Tsukamoto's defuzzification method for the defuzzifier because of its computational simplicity. The reinforcement signal r is calculated as (3).

$$r = \frac{(\mathrm{PM}_0 \cdot \omega_{\mathrm{PM}} + \mathrm{PS}_0 \cdot \omega_{\mathrm{PS}} + \mathrm{NR}_0 \cdot \omega_{\mathrm{NR}} + \mathrm{RS}_0 \cdot \omega_{\mathrm{RS}} + \mathrm{RM}_0 \cdot \omega_{\mathrm{RM}})}{(\omega_{\mathrm{PM}} + \omega_{\mathrm{PS}} + \omega_{\mathrm{NR}} + \omega_{\mathrm{RS}} + \omega_{\mathrm{RM}})} \quad (3)$$

C. The Q-Learning Algorithm

Q-learning learns utility values (Q-values) of state and action pairs. The objective of Q-learning is to estimate Q-values for an optimal policy. During the learning process, learning agent uses its experience to improve its estimate by blending new information into its prior experience.

In general form, Q-learning algorithm is defined by a tuple <S, A, r, p>, where S is the set of discrete state space of ATM networks; A is the discrete action space, which is the feedback signal to traffic sources; r:S×A→R is the payoff function of the agent; p:S×A→Δ(s) is the transition probability map, where $\Delta(s) \in [0,1]$ is the set of probability distributions over state space S.

The basic idea of Q-learning is to find an optimal policy by learning the values of a so-called Q function. $Q(s,a)$ is defined as the expected discounted cumulative payoff that is received by executing action a in state s and following an optimal policy thereafter. Let π be the policy of learning agent, the Q function can be defined as

$$Q^*(s,a) = r(s,a) + \beta \sum_{s' \in S} p(s'|s,a) v(s', \pi^*) . \quad (4)$$

where $\beta \in [0,1)$ is the discount factor, $r(s,a)$ is the reward for taking action a at state s, and $p(s'|s,a)$ is the probability of transiting to the next state s' after taking action a in state s.

If the values of the Q functions are known, the optimal policy π^*, which is always taking an action so as to maximize $Q^*(s,a)$ under any state s, can be found. It is given by

$$\pi^*(s) = \arg\max_{a \in \mathrm{A}} Q^*(s,a) . \quad (5)$$

The problem is then reduced to finding the function $Q^*(s,a)$. Q-learning provides us with a simple updating procedure, in which the learning agent starts with arbitrary initial values of $Q(s,a)$ for all $s \in \mathrm{S}$, $a \in \mathrm{A}$, and updates the Q-values as following

$$Q_{t+1}(s_t, a_t) = (1-\alpha) Q_t(s_t, a_t) + \alpha\left[r_t + \beta \max_a Q_t(s_{t+1}, a)\right] . \quad (6)$$

The parameter $0 \le \alpha < 1$ is called the learning rate and may be decreased over time. The convergence rate is determined by the value of α. If α is small, the convergence rate will be slow but it will easily tend to stabilize. On the other hand, if α is large, the convergence rate will be fast but it will not easily tend to stabilize. The learning rate α should satisfy the following conditions:

1. $\sum_{t=0}^{\infty} \alpha = \infty$, which is required to guarantee that the steps are large enough to eventually overcome any initial conditions.
2. $\sum_{t=0}^{\infty} \alpha^2 < \infty$, which is required to guarantee that the steps are small enough to assure convergence.

It is proven in [12] that the values $Q(s,a)$ estimated using Q-learning converge to the optimal value $Q^*(s,a)$ with probability 1. The values of $Q(s,a)$ are stored in a lookup table, all state-action pairs continue to be visited.

III. CONTROLLER BASED ON SA-Q-LEARNING ALGORITHM

A. The Exploration and Exploitation in Q-Learning

Q-learning algorithm is one of the most rapidly developing reinforcement learning methods in recent years. In Q-learning, a learning agent usually faces a trade-off between exploration and exploitation when choosing an action [13].

Exploitation occurs if the action selection strategy is based purely on current values of the state-action pairs, i.e., when the selection is greedy. This may lead to locally optimal policies, possibly differing from a globally optimal one. In contrast, exploration is the strategy based on the assumption that the agent selects a non-optimal action in the current situation and obtains more knowledge about the problem. This knowledge allows it to neglect the locally optimal policies, and to reach the globally optimal one instead. On the other hand, excessive exploration will drastically decrease the performance of a learning algorithm, and in some cases might be even harmful with respect to the learning results themselves. So, finding the proper balance between exploration and exploitation in Q-learning is one of the key problems in action selection.

A simple strategy proposed to deal with this problem is the ε-greedy (with $0 \le \varepsilon < 1$), with larger ε corresponding to larger

probability of exploration. However, excessive exploration with constant ε value becomes unnecessary after a period of an initial interaction between the learning agent and the environment. Furthermore, it will make the performance of Q-learning algorithm decrease.

Therefore, simulated annealing is introduced in Q-learning algorithm (called SA-Q-learning) to balance exploration and exploitation. Simulating the annealing process of solids, the SA algorithm is one kind of the computational processes resembling nature and has been shown to be an effective approximate algorithm avoiding becoming trapped in a local solution. The SA-Q-learning improves the simple ε-greedy approach by appropriately reducing ε during the learning process. It explores with the probability approaching to 1.0 in the entire environment at beginning. The probability is decreased with learning. At last, the probability approaches to 0, i.e., it will take 0-greedy strategy to select action. This will not only improve the ability of the learning agent to acquire new knowledge, but will also allow the algorithm to avoid performance decrease due to the constant value of ε.

B. The SA-Q-learning algorithm

As the accuracy of the agent's knowledge about the environment increases, the proportion of exploration should decrease. Toward this effect, the SA algorithm is introduced into the action-selection strategy of Q-learning. The resulting algorithm, called SA-Q-learning, is presented below.

Let P be a policy space, the change of any element in a n-tuple $(a_1,a_2,\ldots,a_n)$ corresponding to the transition from one solution to another in policy space P. The value function of a policy is used to evaluate the policy [14].

Let $p \in \mathrm{P}$ be a policy, the value function $V(p)$ of the policy p is the sum of all the Q-values obtained for this policy.

$$V(p) = V\left((a_1, a_2, \cdots, a_n)\right) = \sum_{k=1}^{n} Q^p(s_k, a_k) \tag{7}$$

If the policy $p_1=(a_1,\ldots,a_{i-1},a_i,a_{i+1},\ldots,a_n)$ transits to the policy $p_2=(a_1,\ldots,a_{i-1},b_i,a_{i+1},\ldots,a_n)$, then the difference between their values is

$$V(p_2) - V(p_1) = Q(s_i, b_i) - Q(s_i, a_i) \ . \tag{8}$$

The policy p_1 is considered superior to p_2 if $V(p_1)>V(p_2)$. The value of a policy is compared to the energy of the microcosmic state in solid annealing, and it will be used to decide the probability of the action-selecting, in combination with other parameters, such as the temperature.

The detailed of SA-Q-learning algorithm is shown as the following flowchart (Fig. 3).

1) Generation of initial solution

Generation of the initial solution should ensure the generation of a feasible solution, which can be satisfied by using random number generators distributed evenly from 0 to 1. The following formula makes identical probabilities of the feedback signal (a) falling at any value within the upper limit and the lower limit:

$$a_0 = (a_{\max} - a_{\min}) \cdot \mathrm{random}(0,1) + a_{\min} \tag{9}$$

Initial temperature T_0 should be set great enough and all $Q(s,a)$ values are initiated arbitrarily.

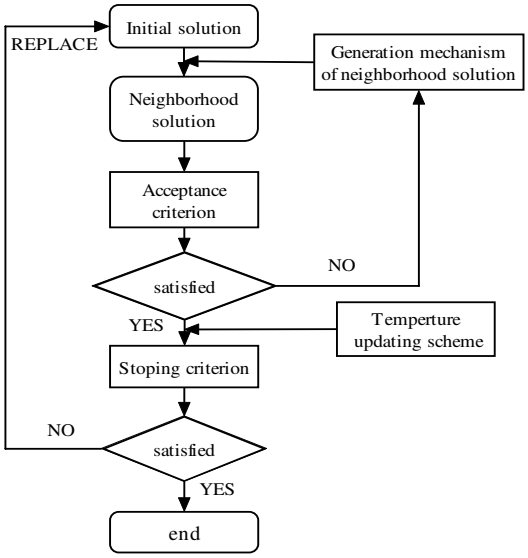

Fig.3. The illustration of the process of SA-Q-learning algorithm

2) Neighborhood solution

Find another solution, namely a' by modifying the present solution a.

The simulated annealing algorithm must generate a new solution in the neighborhood of the present solution. The generating function is described as (10)

$$a' = a + \Delta a \tag{10}$$

$$\Delta a = \mathrm{random}(0,1) \cdot 2\Delta a_{\max} - \Delta a_{\max} \tag{11}$$

3) Acceptance criterion

Execute the Metropolis algorithm under temperature T.

If $Q(s,a') \geq Q(s,a)$, then the trial solution is better than the present one and is thus accepted. If $Q(s,a') < Q(s,a)$, then the new solution a' is accepted with the probability given by

$$\mathrm{Pr} = \exp\left(\left(Q(s,a') - Q(s,a)\right)/T\right) \ . \tag{12}$$

provided that Pr is greater than a random number uniformly distributed over the interval 0 to 1.

The value of $Q(s,a)$ is updated by

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha\left[r + \beta \max_{a'} Q(s',a')\right] \ . \tag{13}$$

4) Cooling schedule

The cooling schedule defines the procedure to reduce the temperature as an equilibrium state is reached.

Although the temperature-dropping criterion can be in general arbitrary, we use the geometric scaling factor criterion to reduce temperature according to the following

$$T_{k+1} = \lambda \cdot T_k \ . \tag{14}$$

The cooling rate λ decides the decreasing velocity of the temperature. When λ is a constant, it is obviously that the different require about the decreasing velocity of the temperature between the high and low temperature region is ignored. In this paper, the cooling rate is adjusted as

$$\lambda = \lambda_0 + \eta \ . \tag{15}$$

In (15), λ_0 is the initial value of the decreasing velocity of

the temperature, and it is set to 0.9. η is a regulating gene, whose value will be increased a little when the temperature T_k is decreased. λ must be always less than 1. It can be seen that when T_k is low, the slower it decreased, the better the system will get stabilization; when T_k is high, the faster it decreased, the smaller the amount of calculation will be.

5) Stopping criterion

The iterative procedure is ended when the stopping criterion is met.

The stopping criterions include the following:

(1) When the cooling temperature is smaller than the final temperature, stop the algorithm.

(2) When the desired number of episodes reaches the preset number, stop the algorithm.

Although ε>0 in Q-learning based on the ε-greedy strategy are often better than those of ε=0, a great number of explorations in case of fixed probability becomes unnecessary and may even be harmful to the system performance as the learning process proceeds. With the introduction of the simulated annealing, SA-Q-learning algorithm eliminates the disadvantage of the probability ε remaining constant, and the exploration will gradually be reduced in par with the dropping temperature.

IV. Simulation Results

In this section, computer simulations are carried out to confirm the validity of the proposed algorithm in ABR traffic control under a variety of networking conditions and loads.

In simulation, VBR voice sources are considered. In ATM networks, VBR service has higher priority than ABR service. In the presence of VBR traffic, ABR capacity becomes a varying quantity. Link bandwidth is first allocated to the VBR service and the remaining bandwidth is given to the ABR service. In simulation, we assume that all ABR sources are greedy and always have data to send.

To simulate the performance of the system, we choose the similar simulation scenarios proposed by Kolarov [15]. It has two switches with finite buffer size of 1680cells each, two groups of ABR sources with each group consisting of five persistent sources and one group of VBR source consisting of four VBR sources as shown in Fig. 4. The desired queue size is set to 50cells. All links have a capacity of 365cells/ms (155Mb/s). Obviously, the link between the two switches is the bottleneck link. The source parameters are chosen as follows: PCR=365cells/ms, ICR=MCR=4cells/ms. Sources in group A start transmission at time t=0, while sources in group B start at time t=5s.

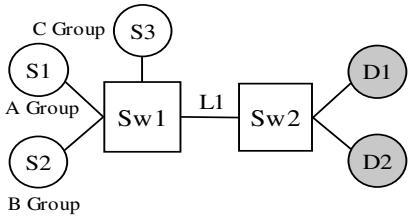

Fig.4. Single bottleneck link simulation model

In order to compare the proposed method with the previous methods, simulations for the same model are carried out by the proposed SA-Q-learning algorithm, 0.1-greedy Q-learning algorithm (ε=0.1 in ε-greedy Q-learning algorithm), and the general Q-learning algorithm. The first successful trial occurs after 35 trials, 83 trials, and 114 trials respectively. It is noted that the training speed increases significantly.

Firstly, we consider the simulation scenario that no VBR traffic flow existing in the networks. Fig.5 shows the rate for each source, when the sources in group A start transmission, the rates converge to stable value 73cells/ms (365/5). When group B sources start transmission, all source rates stabilize around a new equilibrium of 36.5 cells/ms (365/10). From Fig.6, the queue length converges to 50cells which is the buffer set point after about 200ms and there is no overshoot. When group B sources start, the queue is stable after about 100ms with very small overshoot (about 2cells). So it is easy to get the conclusion that the controller can respond to the changes of the network load on time and make the rate of sources and queue stabilize rapidly.

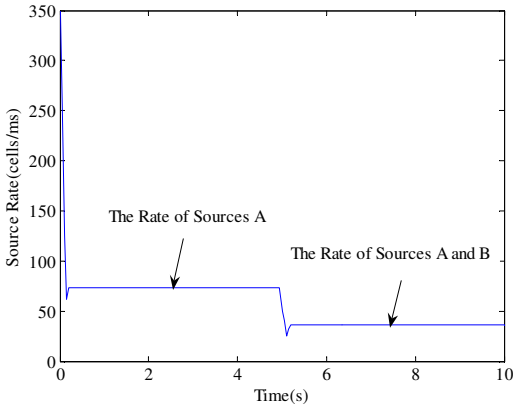

Fig.5. The ABR source rate

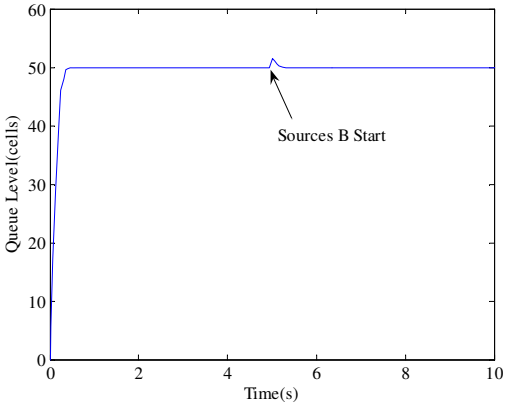

Fig.6. The queue level of buffer

Secondly, we study the performance of proposed controller when the available bandwidth for ABR service continuously changes with time. We add four video MPEG sources at switch1 with a peak rate of 35cells/ms and average rate of 21cells/ms. The MPEG sources have service priority over ABR sources and start transmission over the entire simulation period. The ABR sources are persistent. From Fig.7 after t=0

and t =5s, the rate of ABR sources stabilize around 60cells/ms and 30cells/ms respectively, with little oscillations. This simulation indicates that the controller can reduce sensitivity to the variety of ABR bandwidth.

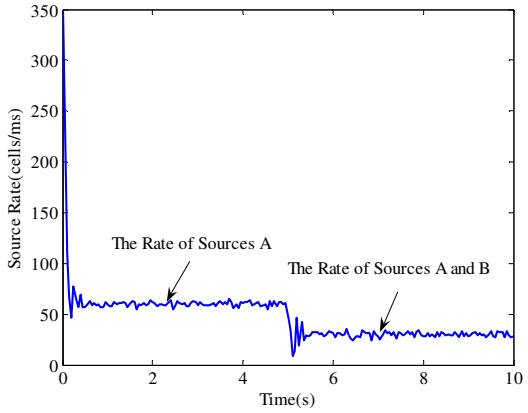

Fig.7. The ABR source rate with VBR existing

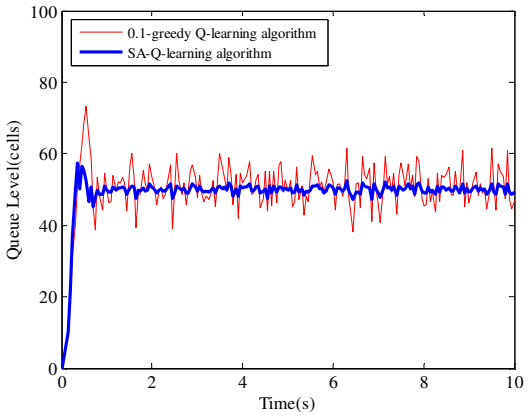

Fig.8. The queue level of buffer with VBR existing

In Fig.8, the performance of queue level in the buffer is considered. For comparison, the proposed SA-Q-learning algorithm and 0.1-greedy Q-learning algorithm is used in the ABR traffic controller and simulated under same condition. Such an illustration is sufficient to draw at least approximate conclusions that the proposed controller has a better performance in ABR traffic control.

V. Conclusion

This paper presents an ABR traffic controller based on Q-learning algorithm conjunction with simulated annealing. Because of the interaction with the environment, the Q-learning algorithm has good performance in the traffic control of ATM networks. The simulated annealing is introduced to deal with the relation between exploration and exploitation in searching for an optimal action. It is seen that the proposed controller indeed performs an efficiently traffic control, and is capable of learning the system behavior. The simulation results show that the proposed controller is superior to the general Q-learning and the random learning ε-greedy Q-learning with respect to the performance of ATM networks.

References

[1] ATM Forum Technical Committee TMWG. ATM Forum Traffic Management Specification Version 4.0, af-tm-0056.000, 1996.

[2] S. Chan, M. Zukerman, E. W. M. Wong, K. T. Ko, E. Yeung and B. Wydrowski, "A congestion control framework for available bit rate service in ATM networks," *International Journal of Communication Systems*, vol. 15, no. 4, pp. 341-357, 2002.

[3] T. Jayasree, and Jagannathan, "Predictive congestion control of ATM networks: Multiple sources/single buffer scenario," *Automatica*, vol. 38, no. 5, pp. 815-820, 2002.

[4] S. J. Lee, and C. L. Hou, "Neural-fuzzy system for congestion control in ATM networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 1, pp. 2-9, 2000.

[5] E. Gnerin, I. W. Habib, S. Palazzo, and C. Douligeris, "Intelligent techniques in high speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 2, pp. 145-155, 2000.

[6] T. Ren, G. M. Dimirovski, and Y. W. Jing, "ABR Traffic Control over ATM Network Using Fuzzy Immune-PID Controller," in *Proceedings of the 2006 American Control Conference*, Minneapolis, Minnesota, USA, June 14-16, 2006, pp. 4876-4881.

[7] A. Chatovich, S.Okug, and G. Dundar, "Hierarchical neuro-fuzzy call admission controller for ATM networks," *Computer Communications*, vol. 24, pp. 1031-1044, 2001.

[8] M. C. Hsiao, S. W. Tan, K. S. Hwang, and C. S. Wu, "A reinforcement learning approach to congestion control of high-speed multimedia networks," *Cybernetics and Systems*, vol. 36, no. 2, pp. 181-202, 2005.

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

[10] M. Z. Guo, Y. Liu, and J. Malec, "A new Q-learning algorithm based on the metropolis criterion," *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 5, pp. 2140-2143, Oct. 2004.

[11] K. S. Hwang, S. W. Tan, M. C. Hsiao, and C. S. Wu, "Cooperative multiagent congestion control for high-speed networks," *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, vol. 35, no. 2, pp. 255-268, Apr. 2005.

[12] C. J. C. H. Watkins, and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279-292, May 1992.

[13] A. F. Atiya, A. G. Parlos, and L. Ingber, "A reinforcement learning method based on adaptive simulated annealing," in *Proc. of the 46th International Midwest Symposium on Circuits and Systems*, 2003, pp.121-124.

[14] M. G. Ji, Z. H. Jin, and H. W. Tang, "An improved simulated annealing for solving the linear constrained optimization problems," *Applied Mathematics and Computation*, vol. 183, no.1, pp. 251-259, 2006.

[15] A. Kolarov, and G. Ramamurthy, "A control-theoretic approach to the design of an explicit rate controller for ABR service," *IEEE/ACM Trans. on Networking*, vol. 7, pp. 741-753, Oct. 1999.