A Near-Optimal Reinforcement Learning Scheme for Energy Efficient point-to-point Wireless Communications

Charles Pandana and K. J. Ray Liu
Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742
Email: cpandana, kjrliu@glue.umd.edu

***Abstract*— We consider the problem of average throughput maximization per total consumed energy in packetized point-to-point wireless sensor communications. Our study results in an optimal transmission strategy that chooses the optimal modulation level and transmit power while adapting to the incoming traffic rate, buffer condition, and channel condition. We formulate the optimization problem as a Markov Decision Processes (MDP). When the state transition probability of MDP is available, the optimal policy of MDP can be obtained using dynamic programming (DP). Since in practical situation, the state transition probability may not be available when the optimization is done, we propose to learn the near-optimal policy through the reinforcement learning (RL) algorithm. We show that RL algorithm learns policy that achieves almost the same throughput as the optimal one, and the learned policy obtains more than twice average throughput compared to the simple constant signal to noise ratio (CSNR) policy, particularly in high packet arrival rate. Moreover, the learning algorithm is robust in tracking the variation of the governing probability.**

I. Introduction

Recent advances in wireless sensor communications demand a highly energy efficient protocol. In these applications, the traditional low power design, focusing mainly on circuits and systems has been shown inadequate [1]. The stringent energy requirement calls for extremely efficient resource allocation algorithms that maximize the energy efficiency using parameters across different communication layers [2]. The concept of energy awareness requires the communication system to reconfigure the transmission parameters from different communication layers according to its environment [2]. Such a cross-layer optimization can be realized in several ways, one feasible solution is to employ an optimal control agent that interacts with different communication layers and dynamically reconfigures each layer's parameters.

There exist several literatures that focus on the wireless resource management. In [3], power control scheme for packet wireless networks is formulated using dynamic programming (DP). The extension of this work to multi-modal power control is also investigated in [4]. In these two schemes, the power control follows a threshold policy that balances the buffer content and the channel interference. In [5], the DP formulation for power control with imperfect channel estimation is addressed. They show that the dynamic programming solution is better than the standard constant signal to noise ratio (CSNR) approach. Joint optimized bit-rate and delay control for packet wireless networks has also been studied within DP framework [6]. Most of the literatures assume the knowledge of the exact probability model and obtain the optimal solution using DP. In practice, the probability model may not be available when the optimization is being done. This motivates us to develop and investigate a stochastic optimization scheme that learns the optimal policy without knowing the governing probabilistic model. Moreover, the algorithm should be able to track the possible variation in the probability model.

In this paper, we focus on the average throughput maximization per total consumed energy in packetized wireless sensor communications from an optimal control point of view. In particular, we formulate the optimization problem as a Markov Decision Process (MDP) [7]. In point-to-point wireless scenario, the communications take place from one transmitter to one receiver without any interference from other transmitters and the channel is assumed to vary according to some random process. The objective of the optimal control agent is to obtain the best modulation and transmit power to maximize the average throughput per total consumed energy adapting to the packet arrival rate, buffer condition and the channel variation. When the governing probability model is available, the posed problem can be solved using DP. Since in practice, the model may not be available during the optimization, we propose to devise a RL algorithm called Actor-Critic (AC) algorithm to find the optimal policy. We show that the obtained policy is very close to the optimal one. We also compare the learned policy with the simple CSNR policy, and we find that the learned policy achieves more than twice throughput compared to the CSNR policy. Moreover, we demonstrate that the learned policy is robust to the slow variation in the probability model, that is the learned policy is able to track the variation in packet arrival rate.

The rest of this paper is organized as follows. In the next section, we review the MDP and its optimal policy. In section III, we explain the RL algorithm. The throughput maximization per total consumed energy in point-to-point communication is presented in section IV. The performance

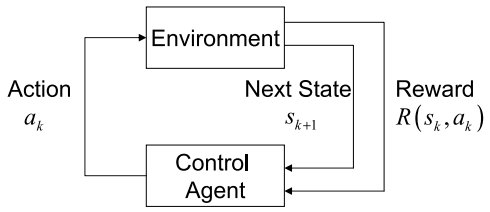

Fig. 1. Interaction between agent and environment in MDP

of RL algorithm is assessed by means of simulations in section V. Finally, conclusions are drawn in Section VI.

II. Markov Decision Process and Dynamic Programming

An MDP [7] is defined as a $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R})$ tuple where $\mathbf{S}$ is the state space that contains all possible states of the system, $\mathbf{A}$ is the set of all possible control actions at each state, $\mathbf{P}$ is the transition function $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$, and $\mathbf{R}$ is the reward function $\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{R}$. The transition function defines a probability distribution over the next state as a function of the current state and the agent's action, i.e. $[\mathbf{P}]_{s_k, s_{k+1}}(a_k) = P_{s_k, s_{k+1}}(a_k)$ specifies the probability of transition from state $s_k \in \mathbf{S}$ to $s_{k+1} \in \mathbf{S}$ under control action $a_k \in \mathbf{A}$. Here, the notation $[\mathbf{A}]_{i,j}$ denotes the element on the i^{th} row and the j^{th} column of matrix $\mathbf{A}$. The transition probability function $\mathbf{P}$ describes the dynamics of the environment as a response to the agent current decision. The reward function specifies the reward incurred at state $s_k \in \mathbf{S}$ under control action $a_k \in \mathbf{A}$. The interaction between the agent and environment in an MDP is illustrated in Figure 1. At time k, the control agent detects $s_k \in \mathbf{S}$ and decides an action $a_k \in \mathbf{A}$. The decision a_k causes the state to evolve from s_k to s_{k+1} with probability $P_{s_k, s_{k+1}}(a_k)$, and reward $R(s_k, a_k)$ corresponding to the agent's action will be obtained.

The solution of the MDP consists of finding the decision policy $\pi : \mathbf{S} \rightarrow \mathbf{A}$ so as to maximize the objective function. In this paper, we focus on the average reward per stage that can be represented as

$$\rho^{\pi}(s_0) = \lim_{n \rightarrow \infty} \frac{1}{n} E_{\pi}\left[\sum_{k=0}^{n-1} R\Big(s_k, \pi(s_k)\Big)\right], \quad s_k \in \mathbf{S}, \pi(s_k) \in \mathbf{A}, \tag{1}$$

where $\rho^{\pi}(s_0)$ is the average reward obtained using decision policy π when the initial state is s_0. Since we are interested in maximizing the average throughput per total consumed energy, this criterion exactly describes our objective function. We note that the expectation operation in (1) is a conditional expectation given a particular policy. The optimal policy is the decision rule that maximizes the average reward per stage over all possible policy π.

By assuming that the Markov chains induced by every policies are irreducible/ergodic, it is shown in [8] that the optimal average reward per stage is independent of the initial state. And for any stationary policy, the corresponding average reward ρ^{π} and relative state value $h^{\pi}(s)$ satisfy the following relation [8]

$$\rho^{\pi} + h^{\pi}(s) = \left[R(s, \pi(s)) + \sum_{s'=1}^{|S|} P_{s,s'}(\pi(s)) h^{\pi}(s')\right], \quad \forall s \in \mathbf{S}. \tag{2}$$

Moreover, the optimal policy is characterized by the Bellman's equation [8]

$$\rho^{*} + h^{*}(s) = \max_{a \in \mathbf{A}(s)} \left[R(s, a) + \sum_{s'=1}^{|S|} P_{s,s'}(a) h^{*}(s')\right], \quad \forall s \in \mathbf{S}, \tag{3}$$

where ρ^{*} is the optimal average reward per stage and $h^{*}(s)$ is known as differential reward or relative state value function for each state s. The Bellman's equation can be solved numerically using dynamic programming (DP) computational methods [8]. We note that solving the Bellman's equation requires the knowledge of state transition probability describing the system. In many practical situation, the state transition probability may not be available during the optimization. In the following sections, we propose to devise the RL algorithm to learn the optimal policy by experiencing the sample path of the process.

III. Reinforcement Learning Algorithm

The main idea in RL algorithm is to update/learn the average reward per stage ρ^{*} and the differential reward $h^{*}(s)$ by means of iterated averaging. We focus on a RL algorithm called Actor-Critic (AC) Algorithm [9] as follow. Let's define the operator $B(h^{\pi}) = R(s, \pi(s)) + \sum_{s'=1}^{|S|} P_{s,s'}(\pi(s)) h^{\pi}(s')$, the relation (2) can be expressed as

$$\begin{aligned} h^{\pi}_{k+1}(s_k) &= B(h^{\pi}(s_k)) - \rho^{\pi}_k \\ \rho^{\pi}_{k+1} &= B(h^{\pi}(s_k)) - h^{\pi}_k(s_k). \end{aligned} \tag{4}$$

The RL algorithm eliminates the need of state transition probability by replacing $B(\cdot)$ operator by $B'(h^{\pi}) = R(s, \pi(s)) + h^{\pi}(s')$, where s' is the next state occurring in the sample path. Obviously, the next state s' will occur according to the probability $P_{s,s'}(\pi(s))$. The RL algorithm learns the state-value function as

$$\begin{aligned} h^{\pi}_{k+1}(s_k) &= (1 - \alpha_k) h^{\pi}_k(s_k) + \alpha_k h^{\pi}_{k+1}(s_k) \\ &= (1 - \alpha_k) h^{\pi}_k(s_k) + \alpha_k (B'(h^{\pi}_k(s_k)) - \rho^{\pi}_k) \\ &= h^{\pi}_k(s_k) + \alpha_k (R(s, \pi(s)) + h^{\pi}_k(s_{k+1}) \\ &\quad - h^{\pi}_k(s_k) - \rho^{\pi}_k) \end{aligned} \tag{5}$$

Similarly, the average reward ρ is updated as

$$\rho^{\pi}_{k+1} = \rho^{\pi}_k + \beta_k [R(s_k, \pi(s_k)) + h^{\pi}_k(s_{k+1}) - h^{\pi}_k(s_k) - \rho^{\pi}_k], \tag{6}$$

The decision/action at each iteration is chosen according to Gibbs softmax method [9], i.e.: action a_k is chosen in state s_k according to probability $Pr(a_k = a | s_k = s) =$

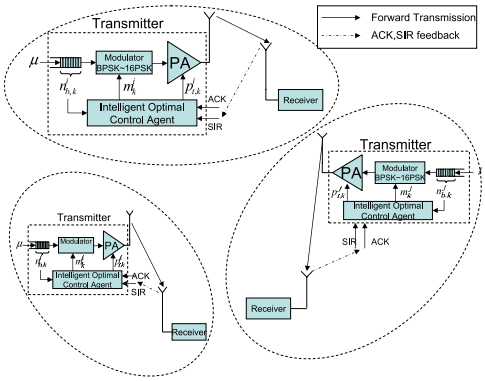

Fig. 2. Interaction of nodes with distributed optimal control agent

$e^{p(s,a)} / \sum_b e^{p(s,b)}$. Whenever the action a_k is chosen at state s_k the preference metric $p(s_k, a_k)$ is updated as

$$p(s_k, a_k) = p(s_k, a_k) + \epsilon_k [R(s_k, a_k) + h_k^\pi(s_{k+1}) - h_k^\pi(s_k) - \rho_k^\pi], \quad (7)$$

The temporal difference, $R(s, \pi(s)) + h_k^\pi(s_{k+1}) - h_k^\pi(s_k) - \rho_k^\pi$ in (5-6) is known as critic, since it criticizes the performance of previous action. Equation (7) is known as actor, since this preference determines the next action. In (5-7), α_k, β_k and ϵ_k determine the learning rate for the average reward, relative value function and decision preference, respectively. Notice that the optimal policy should satisfy the Bellman's equation, the equation (7) effectively maximizes the RHS of (3) that is, the action that results in increasing relative state value function $h_{k+1}(s_k)$ should be prioritized by increasing the preference of choosing that particular action (7). In contrast, the preference of action that results in smaller relative state value function (the temporal difference is negative) will be decreased. The Actor-Critic (AC) algorithm is initialized with $p(s, a) = 0 \quad \forall s \in \mathbf{S}, a \in \mathbf{A}$, this implies that initially every actions have the same probability to be selected. In this initial stage, the algorithm explores all possible actions and decides the best action that leads to the optimal policy. At every time instant, the algorithm will update equation (5)-(7) accordingly.

It is worth to notice that the AC iteration is based on the stochastic approximation algorithm, and typical convergence criteria include that all of the states are visited infinitely often. The convergence behavior of the actor-critic type of learning algorithms can be found in great detail in [10]. Another important aspect is the computational complexity, the complexity of the AC iteration is very low, since only the average reward, relative state value and the decision preference of the visited state and action are updated in every iteration. In the following sections, we formulate the average throughput maximization per total consumed energy in point-to-point wireless sensor networks as an MDP illustrated in Figure 2.

IV. Throughput Maximization in Point-to-Point Communication

We study the average throughput maximization per total consumed energy by choosing the optimal modulation and transmit power adapting to the incoming traffic, transmitter buffer and the channel condition. In point-to-point communication, it is possible to model each of the subsystem using some random processes and the state transition probability for the MDP can be constructed. Having this probability model, we numerically solve the optimal policy using DP algorithm and compare the solution with the near-optimal policy learned by AC algorithm. It is crucial to mention that the AC algorithm does not require any probability model, and the construction of the MDP state transition probability is solely for comparison purpose.

To construct the MDP state transition probability, we model the channel dynamics as a Finite State Markov Channel (FSMC). The basic idea of the FSMC is to partition the channel gain into a finite number of intervals [11]. Each interval represents the channel gain state. In this channel model, the channel transition occurs only at the time slot boundary and the channel is constant during the whole packet transmission. Moreover, the transition only happens between two adjacent states. The channel state transition probability and stationary probability are calculated as in [11].

In the application of wireless sensor networks, the throughput and energy consumption are two critical parameters. We employ the number of packets that successfully transmitted per total energy consumption as our objective function. The total energy consists of energy consumption for the transmission and buffer processing. Including the buffer processing is solely for the Quality of Service (QoS) consideration, that is including the buffer processing energy minimizes the possibility of buffer overflow. Similar to [12], we employ the following reward function

$$R((n_b,\gamma),(m,p_t)) = \begin{cases} \frac{L_b}{L} \frac{R_s \cdot m \cdot S(\Gamma(\gamma,p_t),m)}{L \cdot (p_t + f(n_b))} \times 10^{-3} & \mathrm{n_b \neq 0, p_t \neq 0} \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

The expression (8) has unit packets/mJoule. n_b and γ denote the number of packet queued in the buffer and the channel gain, m and p_t denote the modulation level and transmit power. L_b indicates the number of bits in one packet, L is the number of bits after adding error detecting code. R_s bits/s is the system transmission rate. $f(\cdot)$ models the buffer processing power. $S(\Gamma(\gamma, p_t), m)$ denotes the packet correct reception probability, $\Gamma(\gamma, p_t)$ is the effective link SNR, which can be represented as

$$\Gamma(\gamma, p_t) = \gamma \times \frac{W}{R_s} \frac{p_t \times A_t}{\sigma^2}, \quad (9)$$

where γ is the channel gain variation between the transmitter and receiver and is modelled as FSMC, W denotes the total bandwidth of the transmission, R_s is the system transmission rate, $A_t \propto 1/d^4$ is the attenuation coefficient resulting from path loss, d is the distance between the transmitter and receiver, and σ^2 is the thermal noise power. In this paper, we assume the buffer processing cost is linear function of packets queued in the buffer [3]. Four modulation levels used are BPSK, QPSK, 8PSK and 16PSK. The buffer cost,

packet error probability expression and other parameters are summarized in Table I.

Our optimization objective is to maximize average reward per stage (1), where the reward function is described as in (8). Comparing (1) and (8), we see that the system state is described as the aggregate of buffer content and channel gain, i.e.: $s \equiv (n_b, \gamma)$. The control space consists of modulation level and transmit power, $a \equiv (m, p_t)$. Let's denote the number of packet arrival as $n_a = 0, 1, \ldots$ with probability $p_a(n_a)$. The packet arrival is modelled as Poisson process with mean packet arrival μ. Let's denote the channel transition probability as $p_c(\gamma_k, \gamma_{k+1})$, using FSMC the state transition probability can be determined as in [6] [11]. Suppose the current state is $s_k = (n_{b,k}, \gamma_k)$ and the action taken is $a_k = (m_k, p_{t,k})$. Assuming that the event of packet arrival, correct reception and channel transition are all mutually independent, the MDP state transition probability is determined as follows

1. Transmission failure: $s_{k+1} = (n_{b,k} + n_a, \gamma_{k+1})$

$$P_{s_k, s_{k+1}}(a_k) = p_a(n_a)(1 - S(\Gamma, m_k))p_c(\gamma_k, \gamma_{k+1}) \quad (10)$$

2. Successful transmission: $s_{k+1} = (n_{b,k} + n_a - m_k, \gamma_{k+1})$

$$P_{s_k, s_{k+1}}(a_k) = p_a(n_a)S(\Gamma, m_k)p_c(\gamma_k, \gamma_{k+1}) \quad (11)$$

The overall formulation of MDP has the following interpretation, before a packet transmission, the transmitter is in some state $s_k = (n_{b,k}, \gamma_k)$, this state is obtained from previous transmission's history. The transmitter uses this information to determine the best modulation and power to maximize the average throughput per total consumed energy. At the end of a packet transmission, the transmitter obtains feedback from receiver containing the estimated current channel gain γ_{k+1} and ACK/NACK. If ACK is received then the transmitter will send the following packet at the next transmission. Otherwise, it has to retransmit the packet. This feedback information is used to update transmitter state. The number of successful transmitted packets per the energy consumed in one transmission time (8) serves as the reward.

V. Numerical Results

Based on the above formulation, we construct the simulation using parameters shown in Table I. Given the MDP state transition probability, the optimal solution of the posed MDP problem is solved numerically using policy iteration method in DP [8]. We compare the optimal solution to the policy learned by Actor Critic (AC) algorithm, the AC algorithm parameters are $\alpha = 0.01$, $\beta = 0.0001$ and $\epsilon = 0.01$. For comparison purpose, we also simulate the simple CSNR scheme. In this scheme, the transmitter tries to transmit at the highest throughput possible maintaining predefined link SNRs. The transmitter always chooses the highest modulation possible given the buffer condition, that is it chooses (BPSK, QPSK, 8PSK and 16PSK) when the buffer contains one packet, 2, 3 and ≥ 4 packets, respectively. For each modulation, the transmitter selects power level to achieve predefined link SNRs of (6, 10, 15, 20) for BPSK to 16PSK, respectively. These link SNRs achieve more than 80% of packet correct reception probability.

TABLE I
Simulation parameters

Packet size	$L_b = 64$, $L = 80$
System Parameters	$W = 10$Mhz, $R = 100$kbits/s, $T_p = 0.8$ms, $\sigma^2 = 5 \times 10^{-15}$W
Channel Gain	$f_D = 50$Hz, $\gamma \in= [-8, -6, ..., 8]$ dB, $A_t = 1.916 \times 10^{-14}$
Buffer Cost	$f(n_b) = 0.05(n_b + 4)$ if $n_b \neq \max(n_b)$, $\max(n_b) = 7$, $f(\max(n_b)) = 3$
modulation level	m=1,2,3,4 (BPSK,QPSK,8PSK,16PSK),
Packet success probability	$S(\Gamma(\gamma, p_t), m)$=$(1 - P(\Gamma(\gamma, p_t), m))^L$ $P(\Gamma, m)$=$erfc(\sqrt{\Gamma} * \sin(\frac{\pi}{2^m}))$
Transmit power	$p_t = [0, 0.2, ..., 2]$ Watt
SNR range	$\Gamma = [0, 1, ..., 24]$ dB

Figure 3 shows the average throughput learned by the AC algorithm and the optimal throughput when $\mu = 2.0$. It is obvious that the learned throughput is very close to the optimal one. The corresponding optimal and learned policy for $\mu = 2.0$ are shown in Figure 4. In these figures, the channel is better when the channel gain is larger and the buffer content indicates number of packets queued in the buffer. For the same buffer content, the optimal policy tends to use higher modulation level when the channel is good and lower modulation level when the channel is bad. The agent also tends to select higher power level when the channel is bad to guarantee acceptable throughput. At the same channel gain, as more packets queued in the buffer, the agent becomes more aggressive and attempts higher modulation and power level. This effect is due to including the buffer processing cost in the reward function and the agent tries to balance the transmission energy and buffer processing energy to obtain the maximum average throughput per total expended energy. Moreover, both the optimal DP and the near-optimal AC solution jointly decide the best modulation and transmit power to maximize the average throughput per expended energy.

Figure 5 shows the throughput that can be achieved for the optimal policy, AC learned policy and the simple CSNR policy for various packet arrival rate. It is obvious that the policy learned by the AC algorithm is very close to the optimal policy. Compared to the simple CSNR policy, the AC algorithm obtains twice to three times throughput per total expended energy, hence a higher energy efficiency scheme. It is important to point out that the optimal solution may not be feasible in practical application, since the optimal solution requires the knowledge of channel transition probability and packet arrival probability. The AC algorithm and the simple CSNR algorithm do not require any knowledge of governing probability, but the AC algorithm still obtains a near optimal average throughput. Moreover, the AC algorithm has the ability to track the variation in the governing probability as demonstrated in Figure 6. In this figure, the mean packet arrival rate varies as μ =(0.5, 1.0, 1.5, 2.0, 1.0). Based on

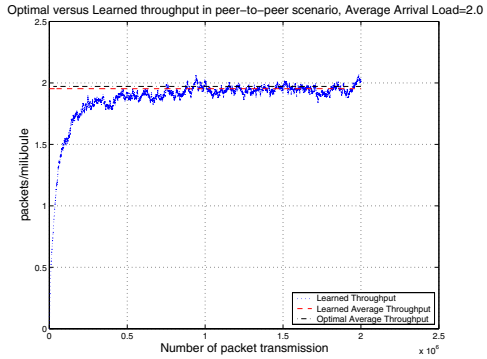

Fig. 3. Learned and optimal average throughput, packet arrival load $\mu = 2.0$

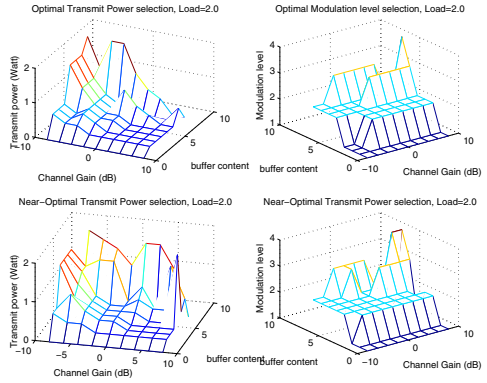

Fig. 4. Learned and optimal policies, packet arrival load $\mu = 2.0$

the sample realization, the AC algorithm adjusts the learned policy adapting to different packet arrival rate. The capability of the AC algorithm to obtain the near-optimal policy and track the variation in the governing probability is due to the ability of algorithm to explore all the possible decisions and select the policy that results in maximum throughput per energy. This exploration is achieved by the Gibbs softmax method used in the actor part of the algorithm.

VI. Conclusion

We formulate the average throughput maximization per total expended energy in point-to-point wireless sensor communications within the MDP framework. We propose to learn the near-optimal policy using the Actor-Critic (AC) algorithm. The learned policy is very close to the optimal one, but without the knowledge of governing probability. Compared to the simple constant link SNR policy, the learned policy can achieve more than two times throughput, especially in high packet arrival rate. Moreover, the AC algorithm is robust in tracking the variation of the governing probability. These advantages come from the fact that the learning algorithm explores all the decisions in the action space and select the policy that maximizes the throughput per unit energy.

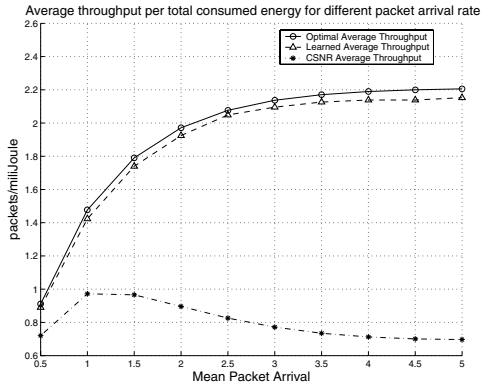

Fig. 5. Average throughput corresponding to different packet arrival load $\mu = 2.0$

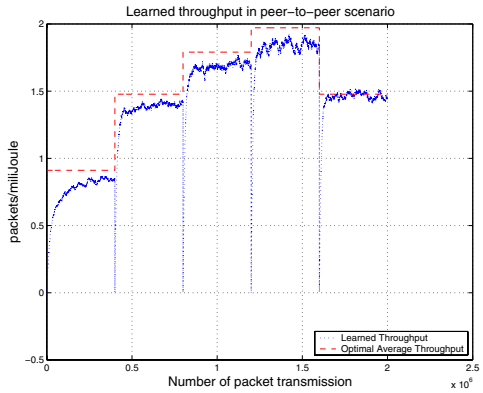

Fig. 6. Learned average throughput when μ=(0.5, 1.0, 1.5, 2.0, 1.0)

References

[1] W. Stark, H. Wang, A. Wrothen, S. Lafortune, and D. Teneketzis, "Low-energy wireless communication network design," *IEEE Wireless Comm.*, pp. 60–72, August 2002.

[2] A. J. Goldsmith and S. B. Wicker, "Design challenges for energy-constrainded ad hoc wireless networks," *IEEE Wireless Comm.*, pp. 8–27, August 2002.

[3] N. Bambos and S. Kandukuri, "Power controlled multiple access (PCMA) in wireless communication networks," *IEEE INFOCOM*, pp. 386–395, 2000.

[4] ——, "Multimodal dynamic multiple access in wireless packet networks," *IEEE INFOCOM*, 2001.

[5] T. Holliday, A. Goldsmith, and P. Glynn, "Wireless link adaptation policies: Qos for deadline constrained traffic with imperfect channel estimates," *IEEE ICC*, 2001.

[6] J. Razavilar, K. J. R. Liu, and S. I. Marcus, "Jointly optimized bit-rate/delay control policy for wireless packet networks with fading channels," *IEEE Trans. on Comm.*, pp. 484–494, March 2002.

[7] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Norwell, MA: Kluwer Academics Publisher, 1999.

[8] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995.

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

[10] V. R. Konda and V. S. Borkar, "Actor-critic-type learning algorithms for markov decision processes," *SIAM Journal on Control and Optimization*, vol. 38, no. 1, pp. 94–123, 1999.

[11] Q. Zhang and S. A. Kassam, "Finite-state markov model for rayleigh fading channels," *IEEE Trans. on Comm.*, pp. 1688–1692, Nov. 1999.

[12] D. J. Goodman and N. B. Mandayam, "Power control for wireless data," *IEEE Personal Communications Magazine*, vol. 7, pp. 48–54, April 2000.