Structure-Aware Stochastic Control for Transmission Scheduling

Fangwen Fu and Mihaela van der Schaar, *Fellow, IEEE*

***Abstract*—In this paper, we consider the problem of real-time transmission scheduling over time-varying channels. We first formulate this transmission scheduling problem as a Markov decision process and systematically unravel the structural properties (e.g., concavity in the state-value function and monotonicity in the optimal scheduling policy) exhibited by the optimal solutions. We then propose an online learning algorithm that preserves these structural properties and achieves ε-optimal solutions for an arbitrarily small ε. The advantages of the proposed online method are given as follows: 1) It does not require *a priori* knowledge of the traffic arrival and channel statistics, and 2) it adaptively approximates the state-value functions using piecewise linear functions and has low storage and computation complexity. We also extend the proposed low-complexity online learning solution to enable prioritized data transmission. The simulation results demonstrate that the proposed method achieves significantly better utility (or delay)–energy tradeoffs compared to existing state-of-the-art online optimization methods.**

***Index Terms*—Delay-sensitive communications, energy-efficient data transmission, Markov decision processes (MDPs), scheduling, stochastic control.**

I. Introduction

WIRELESS systems often operate in dynamic environments where they experience time-varying channel conditions (e.g., fading channel) and dynamic traffic arrivals. To improve the energy efficiency of such systems while meeting the delay requirements of the supported applications, the scheduling decisions (i.e., determining how much data should be transmitted at each time) should be adapted to the time-varying environment [1], [7]. In other words, it is essential to design scheduling policies that consider the time-varying characteristics of the channels and the applications (e.g., backlog in the transmission buffer and priorities of traffic). In this paper, we use optimal stochastic control to determine the transmission scheduling policy that maximizes the application utility given energy constraints.

The problem of energy-efficient scheduling for transmission over wireless channels has intensively been investigated in [1]–[12]. In [1], the tradeoff between the average delay and the average energy consumption for a fading channel is characterized. The optimal energy consumption in the asymptotic large-delay region (which corresponds to the case where the optimal energy consumption is close to the optimal energy consumption with bounded queue length constraint) is analyzed. In [6], joint source–channel coding is considered to improve the delay–energy tradeoff. The structural properties of solutions that achieve the optimal energy–delay tradeoff are provided in [3]–[5]. It is proved that the optimal amount of data to be transmitted increases as the backlog (i.e., buffer occupancy) increases, and the amount of data decreases as the channel conditions degrade. It is also proved that the optimal state-value function (representing the optimal long-term utility starting from one state) is concave in terms of the instantaneous backlog.

We note that the aforementioned solutions are characterized by assuming that the statistical knowledge of the underlying dynamics (e.g., channel-state distribution and packet arrival distribution) is known. When the knowledge is unavailable, only heuristic solutions are provided, which cannot guarantee the optimal performance. For example, to cope with the unknown environment, stability-constrained optimization methods are developed in [8]–[11], where instead of minimizing the queue delay, the focus is on achieving queue stability (i.e., the queue length is always bounded). The order optimal energy consumption is achieved only for asymptotically large queue sizes (corresponding to asymptotic large delays), which is often not suitable for delay-sensitive applications such as video streaming.

Other methods for coping with transmission in an unknown environment rely on online learning algorithms that were developed based on reinforcement learning for Markov decision processes (MDPs), in which the state-value function is learned online, at transmission time [13], [14]. It has been proved that online learning algorithms converge to optimal solutions when all the possible states are infinitely often visited [31]. However, these methods have to learn the state-value function for each possible state, and hence, they require large memory to store the state-value function (i.e., exhibit large memory overhead), and they take a long time to learn (i.e., exhibit a slow converge rate), particularly when the state space is large, as in the considered wireless transmission problem.

In this paper, we consider a transmission model similar to the approach in [1], which consists of a single transmitter and a single receiver on a point-to-point wireless link, where the system is time slotted, and the underlying channel state can be modeled as a finite-state Markov chain [17]. We first formulate the energy-efficient transmission scheduling problem

Manuscript received January 7, 2012; revised June 7, 2012 and July 8, 2012; accepted August 1, 2012. Date of publication August 17, 2012; date of current version November 6, 2012. The review of this paper was coordinated by Prof. V. W. S. Wong.

F. Fu was with the Department of Electrical Engineering, University of California, Los Angeles, CA, 90095 USA. He is now with Intel Corporation, Folsom, CA 95630 USA (e-mail: fangwen.fu@intel.com).

M. van der Schaar is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: mihaela@ee.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2012.2213850

as a constrained MDP problem. We then present the structural properties that are associated with the optimal solutions. In particular, we show that the optimal state-value function is concave in terms of the backlog. Different from the proofs given in [4] and [5], we introduce a postdecision state[1] (which is a "middle" state, in which the transmitter finds itself after a packet transmission but before the new packets' arrivals and new channel realization) and postdecision state-value function, which provide an easier way of deriving the structural results and build connections between the MDP formulation and the queue stability-constrained optimization formulation. In this paper, we show that the stability-constrained optimization formulation is a special case in which the postdecision state-value function has a fixed form that is computed based only on the backlog and without considering the impacts of the time correlation of channel states.

To cope with the unknown time-varying environment, we develop a low-complexity online learning algorithm. Similar to the reinforcement learning algorithm [19], we update the state-value function online when transmitting the data. We approximate the state-value function using piecewise linear functions, which allow us to represent the state-value function in a compact way while preserving the concavity of the state-value functions. Instead of learning the state-value for each possible state, we only need to update the state value in a limited number of states when using piecewise linear approximation, which can significantly accelerate the convergence rate. We further prove that this online learning algorithm can converge to the ε-optimal[2] solutions, where ε is controlled by a user-defined approximation error tolerance. Our proposed method provides a systematic methodology for trading off the complexity of the optimal controller and the achievable performance. As previously mentioned, the stability-constrained optimization uses only the fixed postdecision state-value function (only considering the impacts of backlog), which can achieve the optimal energy consumption in the asymptotic large-delay region, but often exhibits poor performance in the small-delay region, as shown in Section VI. However, our proposed method can achieve ε-optimal performance in both regions. To consider the heterogeneity of the data, we further extend the proposed online learning algorithm to a more complicated scenario, where the data are prioritized and buffered into multiple priority queues.

In this paper, we emphasize the structure-aware online learning for the energy-efficient and delay-sensitive transmission with the following contributions.

- Unlike in the existing literature, we exploit the fact that the unknown dynamics are independent of certain components of the system's state. We utilize this property to perform a batch update on the postdecision state-value function at multiple postdecision states in each time slot. Prior to this paper, it was believed that the postdecision state-based learning algorithm must necessarily be performed one state at a time, because one learns only about the current state being observed and can therefore update only the corresponding component [14]. Importantly, our experimental results illustrate that virtual experience can improve the convergence rate of learning twice faster compared to "one state at a time" postdecision state-based learning. Furthermore, we proposed to update the postdecision state-value function every $T(T \geq 1)$ time slots, thereby allowing us to make a tradeoff between the update complexity and converge rate.
- Instead of updating the state-value function in each state as shown in [13] and [14], we propose a piecewise linear approximation of the postdecision state-value function that exploits the concavity of the postdecision state-value function. We further introduce an approximation error parameter δ to control the number of states to be updated each time, which allows us to perform tradeoffs between the update complexity and the approximation accuracy. Note that this paper is different from the method in [23] in the following two important ways: 1) Unlike in our method, where the value function is directly updated, the method in [24] updates the slopes of the piecewise linear function, thereby requiring the slope to be observable in each state, which is not the case in our problem; and 2) the method in [24] cannot control the accuracy, whereas our method can do this through the approximation error threshold δ.
- When heterogeneous traffic data (e.g., video packets to be transmitted) are considered, priority queues are employed to take into account the unequal importance of the traffic data. An online learning algorithm is proposed to update the postdecision state-value function for each queue, which can take into account the mutual impact among the queues, given their priorities. Our solution considerably differs from the methods in [28] and [29]. In [28], although the precedence between jobs (which is similar to the data priority in our problem) is considered, the update of the state-value function is not presented when the experienced dynamics are unknown. In [29], the multidimensional state-value function is approximated by the span of a set of linear functions, which does not take into account the priorities between the dimensions (corresponding to the different traffic priorities in our formulation).

This paper is organized as follows. Section II formulates the transmission scheduling problem as a constrained MDP problem and presents the methods for solving it when the underlying dynamics are known. Section III introduces the concepts of postdecision state and postdecision state-value function for the considered problem. Section IV presents approximate online learning for solving the MDP problem by exploring the structural properties of the solutions. Section V extends the online learning algorithms to scenarios where the incoming traffic is heterogeneous (i.e., has different priorities). Section VI presents the simulation results, followed by the conclusions in Section VII.

II. Formulating Transmission Scheduling as a Constrained Markov Decision Process

In this paper, we first consider a scheduling problem in which a single user (a transmitter–receiver pair) transmits data over

[1] Similar ideas have been proposed in [14] and [23].

[2] ε-optimal solutions mean that the solutions are within the ε-neighborhood of the optimal solutions.

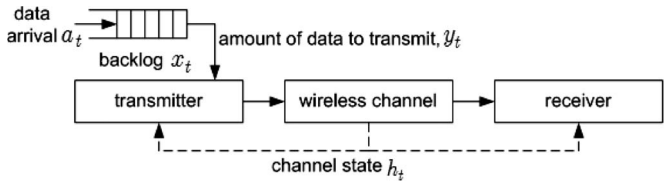

Fig. 1. Transmission scheduling model for a single user.

a time-varying channel in a time-slotted fashion, as shown in Fig. 1. At time slot t $(t = 1, 2, \ldots)$, the user experiences the channel condition (e.g., channel gain) h_t, which is assumed to be constant within one time slot but varying across time slots. The channel condition h_t takes values from a finite set $\mathcal{H}$. To capture the fact that the time-varying channel conditions are correlated with each other, we model the transition of the channel conditions from time slot to time slot as a finite-state Markov chain [17], with the transition probability denoted as $p_c(h_{t+1}|h_t)$. In most deployment scenarios, it is reasonable to assume that the transition process of channel conditions is independent of the supported traffic.

At each time slot t, the following process is performed. Before the transmission, the user observes the transmission buffer with backlog $x_t \in \mathbb{Z}_+$ (representing the number of packets that are waiting for transmission) at the transmitter side [2]. Then, the user transmits the amount of data $y_t \in \mathbb{Z}_+$ $(y_t \leq x_t)$, followed by the data arrival of $a_t \in \mathbb{Z}_+$. For ease of exposition, we also assume, as done in [1], that the traffic arrival a_t is an independent and identically distributed (i.i.d.) random variable[3] with a probability of $p_t(a)$, which is independent of the channel conditions and buffer sizes. Then, the buffer dynamics across time slots is captured by the following expression:

$$x_{t+1} = x_t - y_t + a_t. \tag{1}$$

When the amount of data y_t is transmitted, the user receives the immediate utility of $u(x_t, y_t)$ and incurs the transmission cost $c(h_t, y_t)$. Note that the immediate utility can take negative values. For example, when minimizing the delay, $u(x_t, y_t) = -(x_t - y_t)$, as considered in the simulation in Section VI. One example of transmission cost is the consumed energy. In this paper, we assume that the utility function and the transmission cost function are known *a priori* and satisfy the following conditions.

Assumption 1: $u(x, y)$ is bounded and supermodular, and $-u(x, y)$ is multimodular in (x, y).

Assumption 2: $c(h, y)$ is increasing and multimodular in y for any given $h \in \mathcal{H}$.

The supermodularity[4] in our considered problem means that, when the user has a longer backlog $(x' > x)$, transmitting additional data $(y' - y)$ will lead to higher utility. This assumption is valid for most streaming applications. We note that the assumption of supermodularity on the utility functions is reasonable and has widely been used in previous works [13]. The supermodularity allows us to establish the monotonic structure of the optimal scheduling policy, as shown in Section III.

[3]The method that is proposed in this paper can easily be extended to the case in which the data arrival is Markovian by defining an extra arrival state [6].

[4]A function $f(x, y)$ is supermodular if, for all $x' \geq x$, $y' \geq y$, $f(x', y') - f(x', y) \geq f(x, y') - f(x, y)$ [5].

The multimodularity[5] extends the convexity in the continuous function to the discrete function. It is proved [32] that the extension of $u(x, y)$ is concave, where the extension is performed as follows. It takes the same value as $u(x, y)$ in the integer pair (x, y) and takes values obtained as the corresponding linear interpolation of $u(x, y)$ valued in the four extreme points for other $(x, y) \in \mathbb{R}_+^2$.

The increasing assumption on the transmission cost $c(h, y)$ represents the fact that transmitting more data results in higher transmission cost at the given channel condition h. We introduce the multimodularity on the transmission cost to capture the self-congestion effect [4] of the data transmission.

We define the scheduling policy as a function that maps the current backlog x_t and channel condition h_t into the current action y_t and denote it by $\pi(x_t, h_t)$. We focus on policies that are independent of time and called stationary policies. The objective of the user is to maximize the long-term utility under the constraint on the long-term transmission cost, which is expressed as follows:

$$\begin{aligned} &\max_{\pi \in \Phi} \; E\left[\sum_{t=0}^{\infty} \alpha^t u\left(x_t, \pi(x_t, h_t)\right)\right] \\ &\text{s.t.} \quad E\left[\sum_{t=0}^{\infty} \alpha^t c\left(h_t, \pi(x_t, h_t)\right)\right] \leq \bar{c} \end{aligned} \tag{2}$$

where α is the discount factor in the range [0, 1), Φ is the set of all possible stationary policies, and $\bar{c}$ is the maximum transmission cost. Note that the utility is bounded, and the maximum in (2) is attainable. In this formulation, the long-term utility (transmission cost) is defined as the discounted sum of utility (transmission cost). The discount criteria here put more emphasis on the finite-horizon performance of the scheduling policy with the effective length of the horizon, depending on α. As shown in [33], when $\alpha \to 1$, the effective length is increasing with the order of $-\ln(1-\alpha)/(1-\alpha)$, and the optimal solution to the optimization in (2) approaches the optimal solution to the problem that maximizes the average utility under the average transmission cost constraint, as considered in [1].

The optimization in (2) can be formulated as a constrained MDP. We define the state at time t as $s_t = (x_t, h_t)$, and the action at time t is y_t. Then, the scheduling control is a Markovian system with the state transition probability

$$p\left((x', h')|(x, h), y\right) = p_c(h'|h) p_t\left(x' - (x - y)\right). \tag{3}$$

The long-term utility and transmission cost associated with the policy π are denoted by $U^\pi(s_0)$ and $C^\pi(s_0)$ and can be computed as

$$U^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \alpha^t u\left(x_t, \pi(s_t)\right) | s_0\right] \tag{4}$$

$$C^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \alpha^t c\left(h_t, \pi(s_t)\right) | s_0\right]. \tag{5}$$

[5]A function $f(x, y)$ is multimodular if, for all $(x, y) \in \mathbb{R}_+^2$, (v_1, w_1), $(v_2, w_2) \in F$, $(v_1, w_1) \neq (v_2, w_2)$, $f(x + v_1, y + w_1) + f(x + v_2, y + w_2) \geq f(x, y) + f(x + v_1 + v_2, y + w_1 + w_2)$, where $F = \{(-1, 0), (1, -1)(0, 1)\}$.

Any policy π^* that maximizes the long-term utility under the transmission cost constraint is referred to as the optimal policy. The optimal utility that is associated with the optimal policy is denoted by $U^*_{\bar{c}}(s_0)$, where the subscript indicates that the optimal utility depends on $\bar{c}$. By introducing the Lagrangian multiplier associated with the transmission cost, we can transform the constrained MDP into an unconstrained MDP problem. Based on [15], we know that solving the constrained MDP problem is equivalent to solving the unconstrained MDP and its Lagrangian dual problem. We present this result in Theorem 1 without proof. The detailed proof can be founded in [15].

Theorem 1: The optimal utility of the constrained MDP problem can be computed by

$$\begin{aligned} U^*_{\bar{c}}(s_0) &= \max_{\pi\in\Phi}\min_{\lambda\geq 0} J^{\pi,\lambda}(s_0)+\lambda\bar{c} \\ &= \min_{\lambda\geq 0}\max_{\pi\in\Phi} J^{\pi,\lambda}(s_0)+\lambda\bar{c} \end{aligned} \tag{6}$$

where

$$J^{\pi,\lambda}(s_0)=E\left[\sum_{t=0}^{\infty}\alpha^t\left(u\left(x_t,\pi(s_t)\right)-\lambda c\left(h_t,\pi(s_t)\right)\right)|s_0\right] \tag{7}$$

and a policy π^* is optimal for the constrained MDP if and only if

$$U^*_{\bar{c}}(s_0)=\min_{\lambda\geq 0} J^{\pi^*,\lambda}(s_0)+\lambda\bar{c}. \tag{8}$$

We note that the maximization in the rightmost expression in (6) can be performed as an unconstrained MDP, given the Lagrangian multiplier. Solving the unconstrained MDP is equivalent to solving the Bellman equation, which is presented as follows:

$$\begin{aligned} &J^{*,\lambda}(x,h) \\ &=\max_{\pi\in\Phi}\left[\begin{array}{l} u\left(x,\pi(x,h)\right)-\lambda c\left(h,\pi(x,h)\right)+\alpha\sum\limits_{h'\in\mathcal{H}}p_c(h'|h)\cdot \\ \sum\limits_{x'=x-\pi(x,h)}^{\infty}p_t\left(x'-\left(x-\pi(x,h)\right)\right)J^{*,\lambda}(x',h') \end{array}\right]. \end{aligned} \tag{9}$$

We denote the optimal scheduling policy that is associated with the Lagrangian multiplier λ as $\pi^{*,\lambda}$. The long-term transmission cost that is associated with the scheduling policy $\pi^{*,\lambda}$ is given by

$$C^{\pi^{*,\lambda}}(s_0)=E\left[\sum_{t=0}^{\infty}\alpha^t c\left(h_t,\pi^{*,\lambda}(s_t)\right)|s_0\right]. \tag{10}$$

It was proved in [5] that the long-term transmission cost $C^{\pi^{*,\lambda}}(s_0)$ is a convex function of the Lagrangian multiplier λ. Then, a simple algorithm for finding the optimal Lagragian multiplier λ^* can be found through the following update:

$$\lambda_{n+1}=\max\left(\lambda_n+\gamma_n\left(C^{\pi^{*,\lambda_n}}(s_0)-\bar{c}\right),0\right) \tag{11}$$

where $\gamma_n=1/n$. The convergence to the optimal λ^* is ensured, because $C^{\pi^{*,\lambda}}(s_0)$ is a piecewise convex function of the Lagrangian multiplier λ.

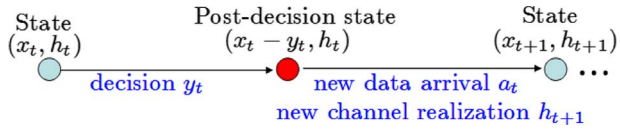

Fig. 2. Illustration of the postdecision state.

III. Postdecision-State-Based Dynamic Programming

In this and the subsequent sections, we will discuss how we can solve the Bellman equations in (9) by exploring the structural properties of the optimal solution for our considered problem. Based on (9), we note that the expectation (over the data arrival and channel transition) is embedded into the term to be maximized. However, in a real system, the distribution of the data arrival and channel transition is often unavailable *a priori*, which makes it computationally impossible to compute the expectation exactly. It is possible to approximate the expectation using sampling, but this approach significantly complicates the maximization.

Similar to [14] and [23], we define

$$V(\tilde{x},\tilde{h})=\sum_{h'\in\mathcal{H}}p_c(h'|\tilde{h})\sum_{x'=\tilde{x}}^{\infty}p_t(x'-\tilde{x})J(x',h') \tag{12}$$

where $\tilde{x}$ $(\tilde{h})$ represents the backlog (channel condition) after scheduling the data but before the new data arrives and new channel state is realized. It is clear that $\tilde{x}_t=x_t-y_t$ and $\tilde{h}_t=h_t$. We refer to the state $\tilde{s}=(\tilde{x},\tilde{h})$ as the postdecision state, because this state is incurred after the scheduling decision has been made at the current time slot. To differentiate the "postdecision" state $\tilde{s}_t$ from the state s_t, we refer to the state s_t as the "normal" state. The postdecision state at time slot t is also illustrated in Fig. 2. If $J(x',h')$ represents the long-term state value starting from the normal state (x',h') at the next time slot $t+1$, then $V(\tilde{x},\tilde{h})$ represents the long-term state value starting from the postdecision state $(\tilde{x},\tilde{h})$ at time slot t. Based on (12), we note that the postdecision state-value function $V(\tilde{x},\tilde{h})$ evaluated at the current time slot is shown as the "weighted average" version of the normal state-value function $J(x',h')$ evaluated at the next slot by taking the expectation over the possible traffic arrivals and possible channel transitions.

The optimal postdecision state-value function $V^{*,\lambda}(\tilde{x},\tilde{h})$ is computed as in (12) by replacing $J(x',h')$ with $J^{*,\lambda}(x',h')$. The Bellman equations in (9) can be rewritten as follows:

$$J^{*,\lambda}(x,h)=\max_{0\leq y\leq x}\left[u(x,y)-\lambda c(h,y)+\alpha V^{*,\lambda}(x-y,h)\right]. \tag{13}$$

The aforementioned equation shows that the normal state-value function at the current time slot is obtained from the postdecision state-value function $V^{*,\lambda}(\cdot,\cdot)$ at the same slot by performing the maximization over the possible scheduling actions. This maximization is referred to as the *foresighted optimization*, because the optimal scheduling policy is obtained by maximizing the long-term utility. Because $-u(x,y)$ and (h,y) are multimodular, it can be proved that $-J^{*,\lambda}(x,h)$ and $-V^{*,\lambda}(x,h)$ are multimodular, and as shown in Section II, their

extension are convex functions. When allowing the scheduling action to take real values (which can be achieved by the mixed policy [5]), the optimization in (13) becomes a convex optimization by replacing the state-value function and postdecision state-value function with their corresponding extended version. Furthermore, the optimal policy $\pi(x,h)$ is nondecreasing as the backlog increases, given the channel state h.

As shown in [14] and [23], introducing the postdecision state and corresponding value functions allows us to perform the maximization without computing the expectation and, hence, without knowledge of the data arrival and channel dynamics. More discussions on postdecision states can be found in [14] and [23]. We can further show that the postdecision state-value function for our considered problem is multimodular (and the extension is concave) in the backlog x in Section IV. Hence, the foresighted optimization in (13) is a one-variable convex optimization. Due to the concavity, we can compactly represent the postdecision state-value functions using piecewise linear function approximations that preserve the concavity and the structure of the problem. As depicted in Fig. 1, we notice that the channel and traffic dynamics are independent of the queue length[6], which enables us to develop a batch update on the postdecision state-value function described in Section IV.

The Bellman equations for the scheduling problem can be solved using value iteration, policy iteration, or linear programming, when the dynamics of the channel and traffic are known *a priori*. However, in an actual transmission system, this information is often unknown *a priori*. In this case, instead of directly solving the Bellman equations, online learning algorithms have been developed to update the state-value functions in real time, e.g., Q-learning [13], [14] and actor-critic learning [19]. However, these online learning algorithms often experience slow convergence rates. In this paper, we develop a low-complexity online learning algorithm by exploiting the structure properties of the optimal solutions, which can significantly increase the convergence rate.

IV. Online Learning Using Adaptive Approximation

In this section, we will first show how the postdecision state-value function can efficiently be updated on the fly when the channel and traffic statistics is unavailable. To deal with the continuous space in backlog, we develop the structural properties of the optimal postdecision state-value function, based on which we will then discuss the approximation of the postdecision state-value function. The approximation allows us to compactly represent and efficiently update the postdecision state-value function.

A. Batch Update for the State-Value Function

As shown in Section III, we still face the following two challenges: 1) Because the queue length (backlog) is infinite, the state space is very large, thereby leading to expensive computation costs and storage overheads; and 2) the channel states and incoming traffic dynamics are often difficult to characterize *a priori*, such that the Bellman equations cannot be solved before the actual traffic transmission.

The Bellman equations provide us with the necessary foundations and principles to learn the optimal state-value functions and optimal policy online. Based on the observation presented in Section III, we note that the expectation is separated from the maximization when the postdecision state is introduced.

We define the postdecision-state-based dynamic programming operator as

$$TV^{\lambda}(\tilde{x},\tilde{h}) = \sum_{h'\in\mathcal{H}} p_c(h'|\tilde{h}) \sum_{x'=\tilde{x}}^{\infty} p_t(x'-\tilde{x}) \max_{0\le y\le x'} \left[u(x',y) - \lambda c(h',y) + \alpha V^{\lambda}(x'-y,h')\right]. \quad (14)$$

Based on this equation, we note that, if we know the postdecision state-value function $V^{\lambda}(\tilde{x},\tilde{h})$, we can perform the foresighted decision (i.e., maximization) without the channel and traffic statistics.

As we know, the statistics of the traffic arrival and channel state transition is not available beforehand. In this case, instead of computing the postdecision state-value function as in (12), we can update online the postdecision state-value function using conventional reinforcement learning [14], [19]. In particular, we can update the postdecision state-value function on the fly as follows.

One-State-per-Time-Slot Update:

We have

$$V^{t,\lambda}(\tilde{x}_{t-1},\tilde{h}_{t-1}) = (1-\beta_t)V^{t-1,\lambda}(\tilde{x}_{t-1},\tilde{h}_{t-1}) + \beta_t J^{t,\lambda}(\tilde{x}_{t-1}+a_{t-1},h_t) \quad (15)$$

where β_t is a learning rate factor [19] that satisfies $\sum_{t=1}^{\infty}\beta_t = \infty$, $\sum_{t=1}^{\infty}(\beta_t)^2 < \infty$, e.g., $\beta_t = 1/t$, and

$$J^{t,\lambda}(x,h) = \max_{0\le y\le x} [u(x,y) - \lambda c(h,y) + \alpha V^{t-1,\lambda}(x-y,h)]. \quad (16)$$

For postdecision states, $(\tilde{x},\tilde{h}) \neq (\tilde{x}_{t-1},\tilde{h}_{t-1}) V^{t,\lambda}(\tilde{x},\tilde{h}) = V^{t-1,\lambda}(\tilde{x},\tilde{h})$.

This method updates only the postdecision state-value function in the postdecision state $(\tilde{x}_{t-1},\tilde{h}_{t-1})$ that is visited at the current time slot, which is referred to the one-state-per-time-slot update. However, in our considered transmission system, we notice that the data arrival probabilities and channel-state transition are independent of the backlog x. In other words, at time slot t, the traffic arrival a_{t-1} and new channel state h_t can be realized at any possible backlog x. Hence, instead of updating the postdecision state-value function only at the state $(\tilde{x}_{t-1},\tilde{h}_{t-1})$, we can update the postdecision state-value function at all states with the same channel state $\tilde{h}_{t-1}$ but different backlogs, which is shown as follows.

Batch Update:

We have

$$V^{t,\lambda}(\tilde{x},\tilde{h}_{t-1}) = (1-\beta_t)V^{t-1,\lambda}(\tilde{x},\tilde{h}_{t-1}) + \beta_t J^{t,\lambda}(\tilde{x}+a_{t-1},h_t) \quad \forall\tilde{x}. \quad (17)$$

[6]If the channel and traffic dynamics depend on the queue length, we can still separate the maximization and expectation. However, the update on the postdecision state-value function is much more complicated and will be investigated in the future.

For postdecision states $(\tilde{x}, \tilde{h})$ with $\tilde{h} \neq \tilde{h}_{t-1}$, $V^{t,\lambda}(\tilde{x}, \tilde{h}) = V^{t-1,\lambda}(\tilde{x}, \tilde{h})$. We refer to the update as the "*batch update*," because it can update the postdecision state-value function $V(\tilde{x}, \tilde{h})$ at all the states $\{(\tilde{x}, \tilde{h}_{t-1}), \forall \tilde{x}\}$.

However, the continuous space in the backlog makes the aforementioned batch update still difficult to be implemented in real systems. In the following section, we try to compactly represent and efficiently update the postdecision state-value function.

B. Online Learning Using Adaptive Approximation

In this section, we present the proposed method for approximating the postdecision state-value function, and we quantify the gap between the approximated postdecision state-value function and optimal postdecision state-value function.

The following theorem shows the structural properties of the optimal postdecision state-value function $V^{*,\lambda}(\tilde{x}, \tilde{h})$.

Theorem 2: With Assumptions 1 and 2, the postdecision state-value function (extended version) $V^{*,\lambda}(\tilde{x}, \tilde{h})$ is a concave function in $\tilde{x}$ for any given $\tilde{h} \in \mathcal{H}$.

Proof: See [27].

In the aforementioned, we derive the structural properties associated with the optimal solutions. It can be proved that the operator T defined in (14) is a maximum norm α-contraction [16], i.e., $\|TV^{\lambda} - TV'^{\lambda}\|_{\infty} \leq \alpha \|V^{\lambda} - V'^{\lambda}\|_{\infty}$ and $\lim_{t\to\infty} T^t V^{\lambda} = V^{*,\lambda}$ for any V^{λ}. Due to the concavity preservation of the postdecision-state-based dynamic programming operator, we choose the initial postdecision state-value function as a concave function in the queue length x and denoted as V_0^{λ}.

We notice that, unlike the traditional Q-learning algorithm, where the state-value function is updated for one state per time slot, our proposed batch update algorithm can update the postdecision state-value function for all the states $\{(\tilde{x}, \tilde{h}_{t-1}), \forall \tilde{x}\}$ in one time slot. The downside of the proposed online learning algorithm is that it has to update the postdecision state-value function $V^{t,\lambda}(\tilde{x}, \tilde{h}_{t-1})$ for all the states of $\{(\tilde{x}, \tilde{h}_{t-1}), \forall \tilde{x}\}$, which is in an infinite space. To overcome this obstacle, we propose to approximate the postdecision state-value function $V^{t,\lambda}(\tilde{x}, \tilde{h})$ using piecewise linear functions, because $V^{t,\lambda}(\tilde{x}, \tilde{h})$ is a concave function. Consequently, instead of updating all the states for the postdecision state-value function, we only update a necessary number of states at each time slot, which is determined by our proposed adaptive approximation method presented in the Appendix. Given the traffic arrival a_{t-1}, new channel state h_t, and the approximated postdecision state-value function $\hat{V}^{t-1,\lambda}$ at time slot $t-1$, we can obtain the optimal scheduling $\pi(x, h_t)$, where $x = \tilde{x} + a_{t-1}$, $\forall \tilde{x}$, and the state-value function $J^{t,\lambda}(x, h_t)$ by replacing the postdecision state-value function $V^{t-1,\lambda}$ in (16) with the approximated postdecision state-value function $\hat{V}^{t-1,\lambda}$. We can then update the postdecision state-value function $V^{t,\lambda}(\tilde{x}, h_{t-1})$ as in (17). However, as previously mentioned, we need to avoid updating the postdecision state-value function in all the states. It has been proved that the postdecision state-value function $V^{t,\lambda}(\tilde{x}, h_{t-1})$ is a concave function. Hence, we propose to approximate the postdecision state-value function $V^{t,\lambda}(\tilde{x}, h_{t-1})$ by a piecewise linear function that preserves the concavity of the postdecision state-value function. In particular, we denote by B_t the largest backlog in the postdecision state that is visited up to the current time slot t. Then, we update the postdecision state-value function as follows:

$$\hat{V}^{t,\lambda}(x, \tilde{h}_{t-1}) = \begin{cases} A^{\delta}_{[0,B_t]} W(x, \tilde{h}_{t-1}); & x \in [0, B_t] \\ W(B_t, \tilde{h}_{t-1}) + k^{W}_{B_t}(x - B_t) & x \in (B_t, \infty) \end{cases} \tag{18}$$

where

$$W(x, \tilde{h}_{t-1}) = (1 - \beta_t)\hat{V}^{t-1,\lambda}(x, \tilde{h}_{t-1}) + \beta_t J^{t,\lambda}(x + a_t, h_t) \tag{19}$$

and $A^{\delta}_{[a,b]}W$ is the approximation operator for any concave function W developed in the Appendix, where the subscript $[a, b]$ emphasizes that the approximation is performed in the range of $[a, b]$, and $k^{W}_{B_t}$ is the slope of the last segment in the piecewise linear approximation $A^{\delta}_{[0,B_t]}$. Then, $A^{\delta}_{[a,b]}W$ is a piecewise linear concave function and satisfies $0 \leq W - A^{\delta}_{[a,b]}W \leq \delta$.

For postdecision states $(\tilde{x}, \tilde{h})$ with $\tilde{h} \neq \tilde{h}_{t-1}$, $\hat{V}^{t,\lambda}(\tilde{x}, \tilde{h}) = \hat{V}^{t-1,\lambda}(\tilde{x}, \tilde{h})$. In the aforementioned equation, we update the postdecision state-value function using δ-controlled piecewise linear approximation in the range $[0, B_t]$ and using the linear function at the range of (B_t, ∞). The largest backlog that has been visited is updated by $B_t = \max(B_{t-1}, \tilde{x}_t)$.

The online learning algorithm is summarized in Algorithm 1.

Theorem 3: Given the concave function operator A_{δ} and the initial piecewise linear concave function $V^{0,\lambda}(\cdot, h)$ for any possible channel state $h \in \mathcal{H}$, if the optimal scheduling policy stabilizes the system, then when applying the online learning algorithm shown in Algorithm 1 every time slot, we have the following conditions.

1) $\hat{V}^{t,\lambda}(\cdot, h)$ is a piecewise linear concave function.
2) $0 \leq V^{*,\lambda}(\cdot, h) - \hat{V}^{\infty,\lambda}(\cdot, h) \leq \delta/(1 - \alpha)$.

Proof: See the Appendix.

Theorem 3 shows that, under the proposed online learning with adaptive approximation, the learned postdecision state-value function converges to the ε-optimal postdecision state-value function, where $\varepsilon = \delta/(1 - \alpha)$, and can be controlled by choosing a different approximation error threshold δ. In Section VI-A, we will show how the approximation error threshold affects the online learning performance.

In Theorem 3, the statement that the optimal scheduling policy stabilizes the system means that the backlog is always finite and, hence, so is B_t.

Algorithm 1: Online learning with adaptive approximation

Initialize: $\hat{V}^{0,\lambda}(\cdot, \tilde{h}) = 0$ for all possible channel states $\tilde{h} \in \mathcal{H}$; postdecision state $\tilde{s}_0 = (\tilde{x}_0, \tilde{h}_0)$; $t = 1$. $B_t = \tilde{x}_0$.

Repeat:

Step 1: Observe the traffic arrival a_{t-1} and new channel state h_t.

Step 2: Compute the normal state (x_t, h_t) with $x_t = \tilde{x}_{t-1} + a_{t-1}$.

Step 3: Batch update the postdecision state-value function, as given in (18).

Step 4: Compute the optimal scheduling policy y_t^* by solving the following optimization and transmit the traffic:

$$J^{t,\lambda}(x_t, h_t) = \max_{0 \le y \le x_t} \left[u(x_t, y) - \lambda c(h_t, y) + \alpha \hat{V}^{t-1,\lambda}(x_t - y, h_t) \right].$$

Step 5: Update the postdecision state $\tilde{s}_t$ with $\tilde{x}_t = x_t - y_t^*$ and $\tilde{h}_t = h_t$.

Step 6: Update the largest backlog by $B_t = \max(B_{t-1}, \tilde{x}_t)$.

Step 7: $t \leftarrow t + 1$.

End

Note that the online learning algorithm with the adaptive approximation shown in Algorithm 1 needs to be performed at each time slot, which may still have high computation complexity, particularly when the number of states to be evaluated is large. To further reduce the computation complexity, we propose to update the postdecision state-value function (using the latest information about channel-state transition and packet arrival) every $T (1 \le T < \infty)$ time slots. It is shown that the online learning performed every T time slots still converges to the ε-optimal solution when the underlying channel-state transition is an aperiodic Markov chain. When the underlying channel-state transition is aperiodic, updating the postdecision state-value function every T time slots will still ensure that every state will be visited infinite times and, hence, will converge to the ε-optimal solution.

In Section VI-A, we also show the impact of choosing different values of T on the delay–energy consumption tradeoff.

C. Stochastic Subgradient-Based Lagrangian Multiplier Update

Based on Section II, we notice that the Lagrangian multiplier is updated using the subgradient, which is given in (11). In this update, the average transmission cost $C^{\pi^*,\lambda_n}(s_0)$ is computed based on the complete knowledge of the data arrival and channel conditions, which is often not available in communication systems. In this case, we use only the realized sample path to estimate the subgradient of the dual problem (i.e., using the stochastic subgradient). In particular, we update the Lagrangian multiplier as follows:

$$\lambda_{k+1} = \left[\lambda_k + \gamma_k \left(\sum_{t=0}^{k} (\alpha)^t c(h_t, y_t) - \bar{c} \right) \right]^+ \tag{20}$$

where $\sum_{t=0}^{k} (\alpha)^t c(h_t, y_t)$ is the approximate stochastic subgradient available at time k, and γ_k is a diminishing step size that satisfies $\sum_{k=1}^{\infty} \gamma_k = \infty$, $\sum_{k=1}^{\infty} (\gamma_k)^2 < \infty$. Then, we can simultaneously update the state-value function and Lagragian multiplier on the fly. Furthermore, to enforce the convergence of the Lagrangian multiplier and the state-value function, γ_k and β_k should also satisfy $\lim_{k\to\infty} \beta_k / \gamma_k = 0$ (see [30] for details). As shown in [30], the key idea behind the convergence proof is characterized as follows. In (17) and (20), the updates of the state-value function $V(s)$ and the Lagrangian multiplier λ are performed using different step sizes. The step sizes satisfy $\lim_{k\to\infty} \beta_k / \gamma_k = 0$, which means that the update rate of the state-value function is faster than the Lagrangian multiplier. In other words, based on the perspective of the Lagrangian multiplier, the state-value function $V(s)$ will approximately converge to the optimal value that corresponds to the current Lagrangian multiplier, because it is updated at the faster time scale. On the other hand, from the perspective of the state-value function, the Lagrangian multiplier appears to be almost constant. These two time-scale updates ensure that the state-value function and the Lagrangian multiplier converge.

D. Comparison With Stability-Constrained Online Learning

In Section IV-B, it is required that the optimal scheduling policy should stabilize the system, which, in some case, is difficult to verify beforehand. Although the system is stabilized under the optimal scheduling policy, the largest backlog B_t may be a large number, thereby leading to heavy computation in the approximation. In this section, we aim at developing a stability-constrained online learning algorithm that will use the constant B for approximation. In particular, we approximate the postdecision state-value function using the piecewise linear function in the range $[0, B]$ and using the Lyapunov function in the range (B, ∞).

In the stability-constrained optimization proposed in [8]–[11], a Lyapunov function is defined for each state (x_t, h_t) as $U(x_t, h_t) = x_t^2$. Note that the Lyapunov function depends only on the backlog x_t. Then, instead of minimizing the tradeoff between the delay and the energy consumption (which is determined by the energy consumption constraint), the stability-constrained optimization minimizes the tradeoff between the Lyapunov drift (between the current state and postdecision state) and energy consumption as

$$\min_{0 \le y_t \le x_t} \lambda c(h_t, y_t) - x_t^2 + (x_t - y_t)^2. \tag{21}$$

Compared to the foresighted optimization in (13), we note that, in the stability-constrained optimization method, the postdecision state-value function is approximated at the whole domain of $[0, \infty)$ by[7]

$$V^{\lambda}(x_t - y_t, h_t) = \left((x_t - y_t) - (x_t - y_t)^2 \right) / \alpha. \tag{22}$$

It has been proved [8] that the chosen Lyapunov function leads to the system stability.

We propose to approximate the postdecision state-value function as follows:

$$\hat{V}^{t,\lambda}(x, \tilde{h}_{t-1}) = \begin{cases} A^{\delta}_{[0,B_t]} W(x, \tilde{h}_{t-1}), & x \in [0, B] \\ \gamma(-x + x^2), & x \in (B, \infty) \end{cases} \tag{23}$$

[7] In [8]–[11], the utility function at each time slot is implicitly defined as $u(x_t, y_t) = -(x_t - y_t)$, which represents the negative value of the postdecision backlog, and the term x_t^2 does not affect the decision.

where $W(x, \tilde{h}_{t-1})$ is given in (19), and $\gamma = k_B^W/(2B-1)$ to ensure that $\hat{V}^{t,\lambda}(x, \tilde{h}_{t-1})$ is a concave function. Now, the difference between our proposed method and stability-constrained method can be summarized as follows.

1) Based on (22), we note that the approximated postdecision state-value function is only a function of the current backlog x_t and the scheduling decision y_t and does not take into account the impact of the channel-state transition and transmission cost in the whole domain $[0, \infty)$. In contrast, we directly approximate the postdecision state-value function based on the optimal postdecision state-value function in the range $[0, B]$, which explicitly considers the channel-state transition and the transmission cost. As shown in the experiments in Section VI-A, the method significantly improves the delay performance in the small-delay region.
2) It has been proved [11], using the stability-constrained optimization method, that the queue length must be larger than or equal to $\Omega(\sqrt{\lambda})$[8] when the energy consumption is within $O(1/\lambda)$[9] of the optimal energy consumption for the stability constraint and that it asymptotically achieves the optimal tradeoff between energy consumption and delay when $\lambda \to \infty$ (corresponding to the large-delay region). However, it provides poor performance in the small-delay region (when $\lambda \to 0$), which is because, as discussed in item 1, the state-value function in this scenario takes into account only the impact of the channel-state transition and cost of the future transmissions. This point is further examined in the numerical simulations presented in Section VI. In contrast, our proposed method can achieve the near-optimal solution in the small-delay region.
3) Furthermore, to consider the average energy consumption constraint, a virtual queue has been maintained to update the Lagrangian multiplier λ in [11]. It can only be shown that this update achieves asymptotical optimality in the large-delay region and results in very poor performance in the small-delay region. Instead, as discussed in Section IV-C, we propose to update λ using stochastic subgradients, which achieves the ε-optimal solution in the small-delay region.

E. Comparison With Other Online Learning Algorithms

In this section, we compare our online learning solution with the algorithms that were proposed in [13], [14], and [23] when applied to the single-user transmission.

In [13], the Q-learning algorithm[10] is presented, where the state–action function (called the Q function) is updated one state at a time, with the constraint that the Q function is submodular. The downsides of this Q-learning method are given as follows: 1) A large table needs to be maintained to store the state–action value function for each state–action pair, which is significantly larger than the state-value function table; and 2) it updates only one entry in the table in each time slot. In [14], an online learning algorithm is presented, which updates the postdecision state-value function. However, the update is still performed one state at a time. The structural properties of the postdecision state-value function are not exploited to speed up the learning. Alternatively, in our proposed online learning with adaptive approximation, we can exploit the structural properties (e.g., concavity) of the postdecision state-value function and approximate it using piecewise linear functions, which require storing the values of only a limited number of postdecision states. It also simultaneously updates the postdecision state-value function at multiple states per time slot and further preserves the concavity of the postdecision state-value function. We show in the simulation results that our proposed online learning algorithm significantly accelerates the learning rate compared to the aforementioned methods.

In [23], an online learning algorithm is presented, which approximates the postdecision state-value function as a piecewise linear function. However, the approximation requires the derivatives of the function to be observable, which is not the case in our problem setting. Moreover, the approximation error in this method is not controllable, which makes it impossible to perform tradeoffs between the computation complexity and the learning efficiency.

The comparison between our method and the existing literature is further summarized in Table I.

TABLE I
COMPARISON OF OUR METHOD WITH THE EXISTING LITERATURE

Methods	Djonin, et al. [5][13]	Salodkar, et al. [14]	Powell, et al. [24]	Proposed
Which state-value function to update	Normal state-value function	Post-decision state- value function	Post-decision state-value function	Post-decision state-value function
How to update state-value function	One state at a time	One state at a time	One state at a time	Batch update every T time slots
Approximation on state-value function	No approximation	No approximation	Piece-wise linear approximation with slope update	Piece-wise linear approximation with function value update
Approximation error control	N/A	N/A	No	Yes
Convergence rate	Slow	Slow	Medium	Fast
Heterogeneous data transmission	Not applicable	Not applicable	Not applicable	Yes

F. Complexity Analysis

In terms of computational complexity, we notice that the stability-constrained optimization performs the maximization shown in (21) once for the visited state at each time slot and the Q-learning algorithm also performs the maximization (finding the optimal state-value function from the state–action value function [23]) once for the visited state at each time slot. In our proposed online learning algorithm, we need to perform the foresighted optimization for the visited state at each time slot. Furthermore, we have to update the postdecision state-value function at the evaluated states. The number of states

[8] $\Omega(\sqrt{\lambda})$ denotes that a function increases at least as fast as $\sqrt{\lambda}$.
[9] In [11], the parameter V is used instead of λ.
[10] Q-learning algorithms may not require knowing the utility functions. However, in this paper, the utility function is assumed to be known.

to be evaluated at each time slot is denoted by n_δ, which is determined by the approximation error threshold δ. If we update the postdecision state-value function every T time slots, then the total number of foresighted optimizations to be performed is, on the average, equal to $1 + n_\delta/T$. Based on the simulation results, we notice that we can often choose $n_\delta < T$, which means that the number of foresighted optimizations to be performed per time slot is less than 2. In other words, our algorithm is comparable to the stability-constrained optimization and Q-learning algorithms in terms of computation complexity.

V. Approximate Dynamic Programming for Multiple Priority Queues

In this section, we consider that the user delivers prioritized data, buffered in multiple queues. The backlog state is then denoted by $\boldsymbol{x}_t = [x_{1,t}, \ldots, x_{N,t}] \in \mathbb{R}_+^N$, where x_{it} represents the backlog of queue i at time slot t, and N is the number of queues. The decision is denoted by $\boldsymbol{y}_t = [y_{1,t}, \ldots, y_{N,t}]$, where y_{it} represents the amount of traffic that is transmitted at time slot t. Similar to the assumptions in Section II, we assume that the immediate utility has the additive form of $u(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^N u_i(x_i, y_i)$, where $u_i(x_i, y_i)$ represents the utility function of queue i, and the transmission cost is given by $c(h, \boldsymbol{y}) = c(h, \sum_{i=1}^N y_i)$. The immediate utility and transmission cost satisfy the following conditions.

Assumption 3: The utility function for each queue satisfies Assumption 1.

Assumption 4: $c(h, y)$ is increasing and multimodular in y for any given $h \in \mathcal{H}$.

Based on Assumptions 3 and 4, we know that $u(\boldsymbol{x}, \boldsymbol{y}) - \lambda c(h, \boldsymbol{y})$ is supermodular in the pair of $(\boldsymbol{x}, \boldsymbol{y})$ and jointly concave in $(\boldsymbol{x}, \boldsymbol{y})$. Similar to the problem with one single queue, the following theorem shows that the optimal scheduling policy is also nondecreasing in the buffer length $\boldsymbol{x}$ for any given $h \in \mathcal{H}$ and the resulted postdecision state function is a concave function.

Theorem 4: Based on Assumptions 3 and 4, the postdecision state-value function $V^{*,\lambda}(\tilde{\boldsymbol{x}}, \tilde{h})$ is a concave function in $\tilde{x}$ for any given $h \in \mathcal{H}$.

Proof: The proof is similar to the proof of Theorem 2 and is omitted here due to space limitations.

Similar to the approximation in the postdecision state-value function for the single-queue problem, the concavity of the postdecision state-value function $V^{*,\lambda}(\tilde{\boldsymbol{x}}, \tilde{h})$ in the backlog $\tilde{\boldsymbol{x}}$ enables us to approximate it using multidimensional piecewise linear functions [22]. However, approximating a multidimensional concave function has high computation complexity and storage overhead due to the following reasons.

1) To approximate an N-D concave function, if we sample m points in each dimension, the total number of samples to be evaluated is m^N. Hence, we need to update m^N postdecision state-values in each time slot and store the m^N postdecision state-values. We notice that the complexity still exponentially increases with the number of queues.
2) To evaluate the value at postdecision states that are not the sample states, we require N-D interpolation, which is often required to solve a linear programming problem [22]. Given the postdecision state-values at these sample points, computing the gap requires also solving the linear program. Hence, the computation in solving the maximization for the state-value function update still remains complex.

However, we notice that, if the queues can be prioritized, this can significantly simplify the approximation complexity, as discussed next. First, we formally define the prioritized queues as follows.

Definition (Priority Queue): Queue j has a higher priority than queue k (denoted as $j \triangleleft k$) if the following condition holds:

$$u_j(x_j, y_j + \triangle y) - u_j(x_j, y_j) > u_k(x_k, y_k + \triangle y) - u_k(x_k, y_k),\ \forall x_j, x_k, y_j + \triangle y \le x_j, y_k + \triangle y \le x_k.$$

The priority definition in the aforementioned expression shows that transmitting the same amount of data from queue j always gives us higher utility than transmitting data from queue k. One example is $u(\boldsymbol{x}, h, \boldsymbol{y}) = w_1 \min(x_1, y_1) + w_2 \min(x_2, y_2)$, with $w_1 = 1, w_2 = 0.8$. It is clear that queue 1 has higher priority than queue 2. In the following theorem, we will show how the prioritization affects the packet scheduling policy and the state-value function representation. We assume that the N queues are prioritized and $1 \triangleleft 2 \cdots \triangleleft N$. The following theorem shows that the optimal scheduling policy can be found queue by queue and the postdecision state-value function can be presented using N 1-D concave functions.

Theorem 5: The optimal scheduling policy at the normal state $(\boldsymbol{x}, h)$ and postdecision state-value function at the postdecision state $(\tilde{\boldsymbol{x}}, \tilde{h})$ can be solved as follows.

1) The optimal scheduling policy for queue i is obtained by solving the foresighted optimization as

$$y_i^* = \arg \max_{0 \le y_i \le x_i} \left\{ u_i(x_i, y_i) - \lambda c\left(h, y_i + \sum_{j=1}^{i-1} y_j^*\right) + \alpha V_i^{*,\lambda}(x_i - y_i, h) \right\}, \forall i. \quad (24)$$

2) The optimal scheduling policy satisfies the condition of $(x_i - y_i^*) y_j^* = 0$ if $i \triangleleft j$.
3) The postdecision state-value function $V_i^{*,\lambda}(\tilde{x}_i, \tilde{h})$, $\forall i$, is a 1-D concave function for fixed h and is computed as

$$V_i^{*,\lambda}(\tilde{x}_i, \tilde{h}) = \sum_{a_1,\ldots,a_i} \sum_{h'} \prod_{j=1}^{i} p_t(a_j) p_c(h'|h) \max_{0 \le y_i \le \tilde{x}_i + a_i} \times \left\{ \sum_{j=1}^{i-1} u_i\left(a_j, z_j^*\right) + u_i(\tilde{x}_i + a_i, y_i) - \lambda c\left(h', y_i + \sum_{j=1}^{i-1} z_j^*\right) + \alpha V_i^{*,\lambda}(\tilde{x}_i + a_i - y_i, h') \right\} \quad (25)$$

where

$$z_i^* = \arg \max_{0 \le z_i \le a_i} \left\{ u_i(a_i, z_i) - \lambda c\left(h', z_i + \sum_{j=1}^{i-1} z_j^*\right) + \alpha V_i^{*,\lambda}(a_i - z_i, h') \right\}. \quad (26)$$

Proof: The key idea is to apply the backward induction from the higher priority queue to the lower priority queue by taking advantage of the fact that the lower priority data will not affect the transmission of the higher priority data. We refer the readers to [27] for more details.

In Theorem 5, statements 1 and 2 indicate that, when queue i has a higher priority than queue j, the data in queue i should first be transmitted before transmitting any data from queue j. In other words, if $y_j^* > 0$ (i.e., some data are transmitted from queue j), then $x_i = y_i^*$, which means that all the data in queue i have been transmitted. If $x_i > y_i^*$ (i.e., some data in queue i are not yet transmitted), then $y_j^* = 0$, which means that no data are transmitted from queue j. When transmitting the data from the lower priority queue, the optimal scheduling policy for this queue should be solved by considering the impacts of higher priority queues through the convex transmission cost, as shown in (24). We further notice that, to obtain the optimal scheduling policy, we only need to compute N 1-D postdecision state-value functions, each of which corresponds to one queue.

Based on the aforementioned discussion, we know that, because $i \lhd j$, the data in queue i must be transmitted earlier than the data in queue j. Hence, to determine the optimal scheduling policy y_i^*, we require only the postdecision state-value function $V^{*,\lambda}((0, \ldots, 0, \tilde{x}_i, \ldots, \tilde{x}_n), h)$. We further notice that the data at the lower priority queues $(i \lhd j)$ do not affect the scheduling policy for queue i. In Theorem 5, statement 3 indicates that $V_i^{*,\lambda}(\tilde{x}_i, \tilde{h})$ is updated by setting $x_k = 0$, $k \lhd i$. Note that the update of $V_i^{*,\lambda}(\tilde{x}_i, \tilde{h})$ is 1-D optimization and $V_i^{*,\lambda}(\tilde{x}_i, \tilde{h})$ is concave. Hence, we can develop online learning algorithms with adaptive approximation for updating $V_i^{*,\lambda}(\tilde{x}_i, \tilde{h})$. The online learning algorithm is illustrated in [27].

Compared to the priority queue systems [26], where there is no control on the amount of data to be transmitted at each time slot, our algorithm is similar in the transmission order, i.e., always transmitting the higher priority data first. However, our proposed method further determines how much should be transmitted at each priority queue in each time slot.

VI. Simulation Results

In this section, we perform numerical simulations to highlight the performance of the proposed online learning algorithm with adaptive approximation and compare it with other representative scheduling solutions.

A. Transmission Scheduling With One Queue

In this simulation, we consider a wireless user who transmits traffic data over a time-varying wireless channel. The objective is to minimize the average delay while satisfying the energy constraint. Due to Little's theorem [18], it is known that minimizing the average delay is equivalent to minimizing the average queue length (i.e., maximizing the negative queue length and $u(x, y) = -(x - y)$). The energy function for transmitting the amount of y (in bits) traffic at the channel state h is given by $c(h, y) = \sigma^2(2^y - 1)/h^2$, where σ^2 is the variance of the white Gaussian noise [18]. In this simulation, we choose $\overline{h}^2/\sigma^2 = 0.14$, where $\overline{h}$ is the average channel gain. We divide the entire channel gain range into eight regions, each of which is represented by a representative state. The states are presented in Table II. The number of incoming data falls into the Poisson distribution [26] with a mean of 1.5 Mb/s. To obtain the time average delay as computed in [11], we choose $\alpha = 0.95$. The transmission system is time slotted with a time slot length of 10 ms.

TABLE II
Channel States Used in the Simulation

Channel gain (h^2 / σ^2) regions	Representative states
(0, 0.0280], (0.0280, 0.0580], (0.0580, 0.0960], (0.0960, 0.1400], (0.1400, 0.1980], (0.1980, 0.2780], (0.2780, 0.4160], (0.4160, ∞]	0.0131, 0.0418, 0.0753, 0.1157, 0.1661, 0.2343, 0.3407, 0.6200

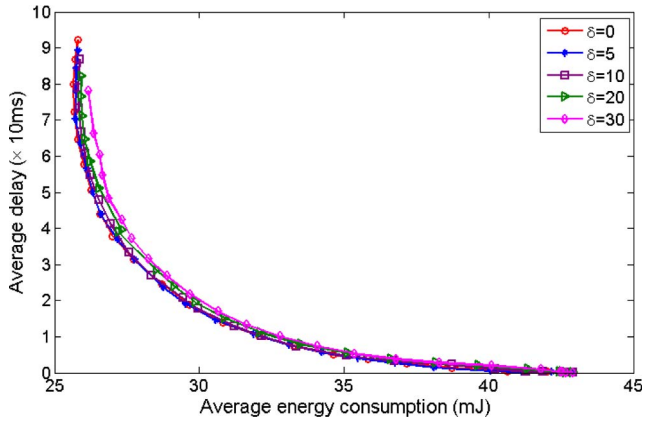

Fig. 3. Delay–energy tradeoff with different approximation error thresholds.

1) Complexity of Online Learning With Adaptive Approximation: In this simulation, we assume that the channel-state transition is modeled as a finite-state Markov chain and the transition probability can be computed as shown in [17]. We approximate the postdecision state-value function using piecewise linear functions in the range $[0, B]$, with $B = 500$ and using Lyapunov functions in the range (B, ∞). As discussed in Section IV-B, by choosing a different approximation error threshold δ, we can approximate the postdecision state-value function by evaluating a different number of states and at different accuracy. The simulation results are obtained by running the online learning algorithm for 10 000 time slots. Fig. 3 shows the delay–energy tradeoff obtained by the online learning algorithms with different approximation error thresholds, and Table III illustrates the corresponding number of states that need to be evaluated. It is easy to see that, when the approximation error threshold δ increases from 0 to 30 (note that $\delta = 0$ indicates that the postdecision state-value functions are

TABLE III
NUMBER OF STATES THAT ARE UPDATED AT EACH TIME SLOT

	$\delta = 0$	$\delta = 5$	$\delta = 10$	$\delta = 20$	$\delta = 30$
#states to update	500	14	10	7	5

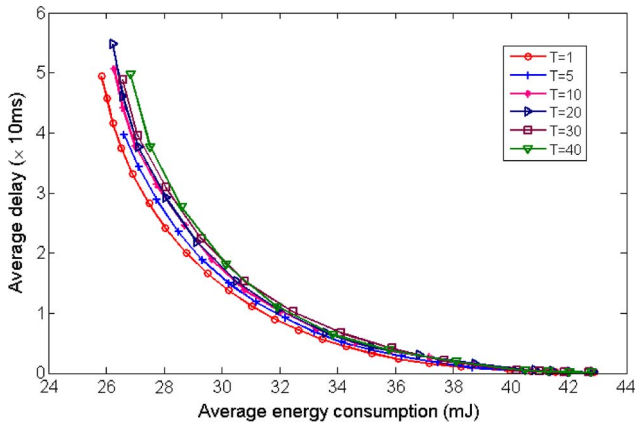

Fig. 4. Delay–energy tradeoff obtained with different update frequencies.

evaluated at each integer backlog and the obtained solution is Pareto optimal), the tradeoff curve further moves from the Pareto front, which means that, to obtain the same delay, the learning algorithm with higher approximation error threshold increases the energy consumption. We notice that the energy increase is less than 5%. However, the number of states required to evaluate at each time slot is significantly reduced from 500 to 5.

To further reduce the computation complexity, instead of updating the postdecision state-value function every time slot, as previously performed, we update the postdecision state-value function every T time slots, where $T = 1, 5, 10, 20, 30,$ and 40. The delay–energy tradeoffs obtained by the online learning algorithm with adaptive approximation are depicted in Fig. 4, where $\delta = 10$. On one hand, we note that, when T increases from 1 to 40, the amount of energy consumed to achieve the same delay performance is increased. However, the increase is less than 10%. On the other hand, based on Table II, we note that we only need to update ten states at each time slot when $\delta = 10$. If we update the postdecision state-value function every $T = 40$, then, on the average, we only need to update 1.25 states per time slot, which significantly reduces the learning complexity.

2) Convergence of Online Learning Algorithm: In this section, we verify the convergence of the proposed online learning algorithm, which uses adaptive approximations of the postdecision state-value function update and stochastic subgradients for the Lagrangian multiplier update. The simulation setting is the same as in Section VI-A1, with $\delta = 5$ and $T = 10$. Fig. 5 shows the convergence of the experienced average delay under different energy constraints. It confirms that the proposed method converges to the ε-optimal solution after 10 000 time slots. We further compare our online learning approach with the approach that was proposed in [14] and present the simulation results in Fig. 6. It is shown that our proposed approach can

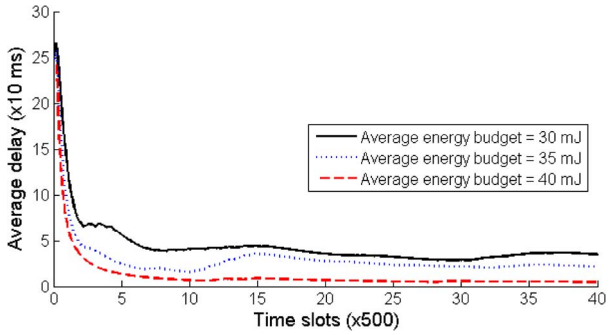

Fig. 5. Convergence of the average delay under different energy budget.

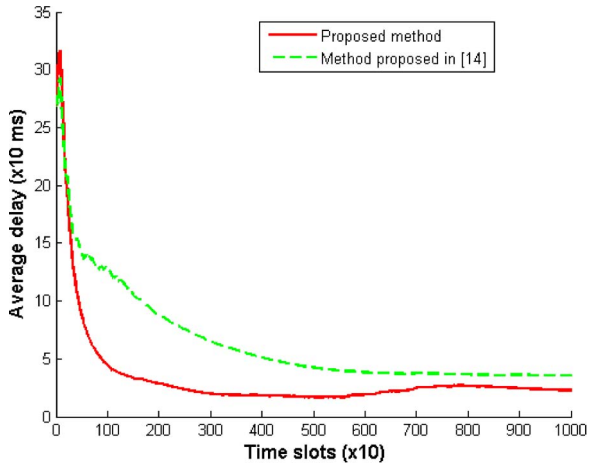

Fig. 6. Convergence rate comparison between the proposed approach and the approach in [14]

achieve the optimal delay twice faster (our approach only needs 3000 time slots, and the approach in [14] needs 7000 time slots) than the approach in [14]. This improvement is because we can exploit the structural properties of the postdecision state-value function and update it at multiple states at a time, whereas the approach in [14] updates only the postdecision state-value function one state at a time without utilizing the structural properties.

3) Comparison With Other Representative Methods: In this section, we compare our proposed online learning algorithm with other representative methods. In particular, we first compare our method with the stability-constrained optimization method that was proposed in [11] for single-user transmission, which approximates the postdecision state-value functions using Lyapunov functions in the whole range of backlog, i.e., $[0, \infty)$. We consider the following two scenarios: 1) i.i.d. channel gain, which is often assumed by the stability-constrained optimization; and 2) Markovian channel gain, which is assumed in this paper. In this simulation, the tradeoff parameter (Lagrangian multiplier) λ is updated through a virtual queue in the stability-constrained optimization and through the stochastic subgradient method in our proposed method. In our method, $\delta = 10$, and $T = 10$.

Figs. 7 and 8 show the delay–energy consumption tradeoffs when the data are transmitted over these three different channels. Based on these figures, we note that our proposed method outperforms the stability-constrained optimization at both the

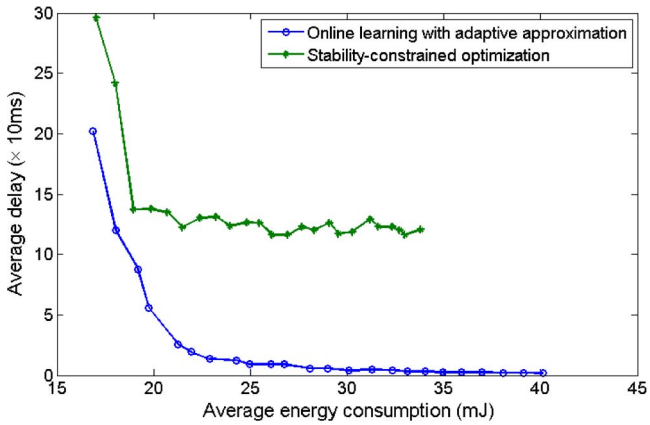

Fig. 7. Delay–energy tradeoff when the underlying channel is i.i.d.

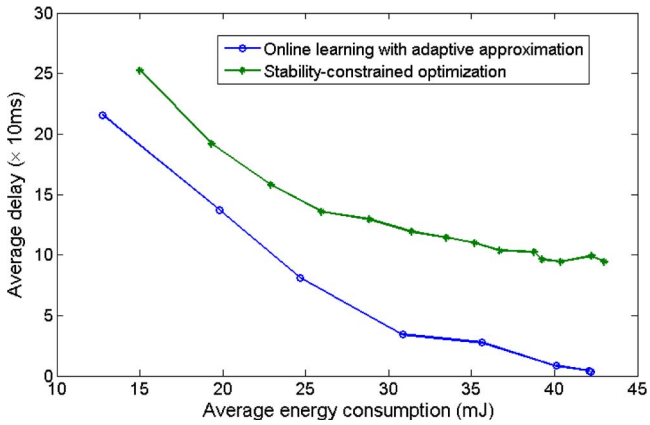

Fig. 8. Delay–energy tradeoff when the underlying channel is Markovian.

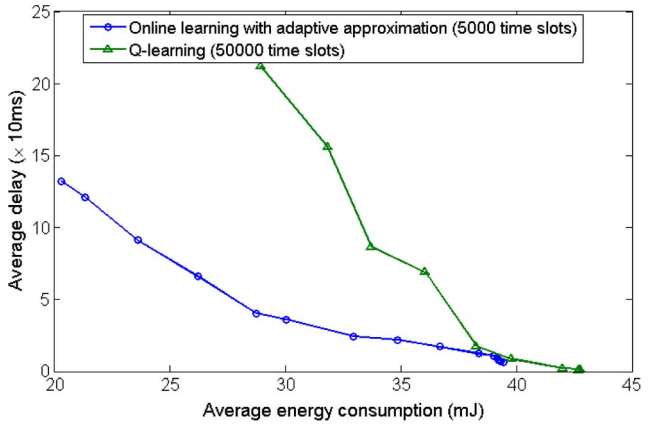

Fig. 9. Delay–energy tradeoff obtained by different online learning algorithms when the channel is Markovian.

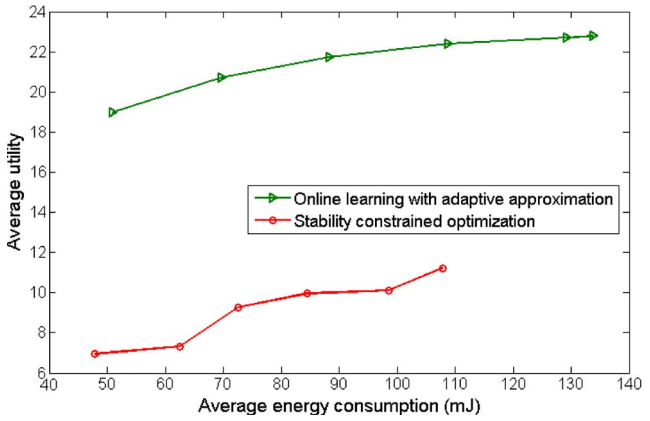

Fig. 10. Utility–energy tradeoff of the prioritized traffic transmission when the channel is Markovian.

large-delay region ($\geq$15) and the small-delay region. We also note that, in the large-delay region, the difference between our method and the stability-constrained optimization becomes small, because the stability-constrained optimization method asymptotically achieves the optimal energy consumption, and our method is ε optimal. However, in the small-delay region, our method can significantly reduce the energy consumption for the same delay performance. We further note that the stability-constrained method could not achieve zero delay (i.e., data are processed once they enter into the queue), even if the energy consumption increases. This case is because the stability-constrained optimization method minimizes only the energy consumption at the large-delay region and does not perform optimal energy allocation at the small-delay region, because the queue length is small. In contrast, our proposed online learning can take care of both regions by adaptively approximating the postdecision state-value functions.

We then compare our proposed method with the Q-learning algorithm as proposed in [14]. In this simulation, we transmit the data over the Markovian channel. In the Q-learning algorithm, the postdecision state-value function is updated for one state per time slot. Fig. 9 shows the delay–energy tradeoffs. The delay–energy tradeoff of our proposed method is obtained by running our method for 5000 time slots. The delay–energy tradeoff of the Q-learning algorithm is obtained by running Q-learning algorithm for 50 000 time slots. It is shown in Fig. 9 that our proposed method outperforms the Q-learning algorithm even when our algorithm learns only more than 5000 time slots and the Q-learning algorithm learns more than 50 000 time slots. Hence, our method significantly reduces the amount of time to learn the underlying dynamics (i.e., experiencing faster learning rate) compared to the Q-learning algorithm.

B. Transmission Scheduling With Multiple Priority Queues

In this section, we consider that the wireless user schedules the prioritized data over a time-varying wireless channel. The channel configuration is the same as shown in Section VI-A. The wireless user has two prioritized classes of data to be transmitted. The utility function is given by $u(x,h,y) = w_1 \min(x_1, y_1) + w_2 \min(x_2, y_2)$, where $w_1 = 1.0$ and $w_2 = 0.8$ represent the importance of the data at classes 1 and 2, respectively. Thus, we have $1 \triangleleft 2$. Fig. 10 illustrates the utility–energy tradeoffs obtained by the proposed online learning algorithm and the stability-constrained optimization method. Fig. 11 shows the corresponding delay–energy tradeoffs experienced by each class of data. Fig. 10 shows that, at the same energy consumption, our proposed algorithm can achieve utility that is 2.2 times higher than the utility obtained by the stability-constrained optimization method. Note that class 1 has less delay than class 2, which is demonstrated in Fig. 11,

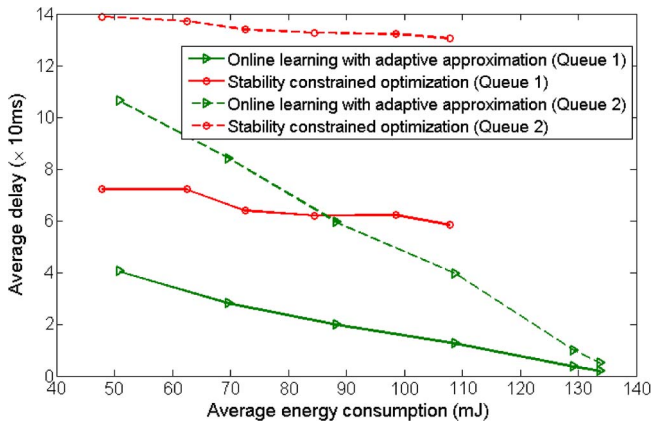

Fig. 11. Delay–energy tradeoff of each class in the prioritized traffic transmission when the channel is Markovian.

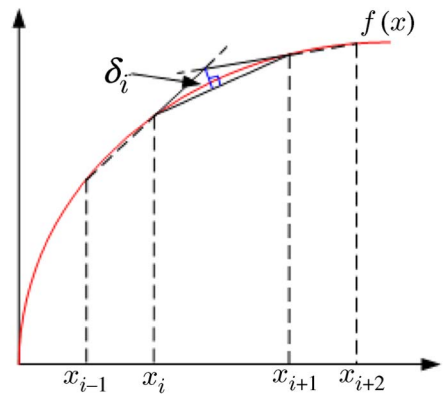

Fig. 12. Lower and upper bounds of the concave function $f(x)$ in the range of $[x_i, x_{i+1}]$.

because class 1 has higher priority. Fig. 11 also shows that the delay is reduced by 50%, on the average, for each class in our method compared to the stability-constrained optimization method. This improvement is because our proposed method explicitly considers the time-correlation in the channel-state transition and the priorities in the data.

VII. Conclusion

In this paper, we first established the structural results of the optimal solutions to the constrained MDP formulation of the transmission scheduling problems. Based on these structural properties, we propose to adaptively approximate the postdecision state-value function using piecewise linear functions, which can preserve the structural properties. Furthermore, this approximation allows us to compactly represent the postdecision state-value functions and learn them with low complexity. We prove that the online learning with adaptive approximation converges to the ε-optimal solutions, the size of which is controlled by the predetermined approximation error. We extend our method to the heterogeneous data transmission, in which the incoming traffic is prioritized.

One extension of our method is multiuser transmission scheduling, in which users transmit the delay-sensitive data over the shared wireless channels. Similar to the way that we formulate the single-user transmission scheduling, the multiuser transmission scheduling can also be formulated as a constrained Markov decision problem. We can also show that the corresponding postdecision state-value function is a concave function of the users' backlog. However, the postdecision state-value function is a multidimensional nonseparable function, which is difficult to update over time. One important challenge is how we can efficiently update the postdecision state-value function in a distributed manner in decentralized wireless networks. In [26], we present several preliminary results on how we can decompose the postdecision state-value function into multiple single-user postdecision state-value functions and update them in a distributed fashion. This approach can form a starting point for solving the multiuser transmission scheduling in a decentralized wireless environment. Another extension of our method is heterogeneous data transmission, in which the data have different delay-deadlines, priorities, and dependencies [25].

Appendix

Approximating the Concave Function: In this section, we present a method for approximating a 1-D concave function. Consider a concave and increasing function $f : [a, b] \to \mathbb{R}$ with n points $\{(x_i, f(x_i))|i = 1, \ldots, n\}$ and $x_1 < x_2 < \cdots < x_n$. Based on these n points, we can give the lower and upper bounds on the function f. It is well known that the straight line through the points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$ for $i = 1, \ldots, n-1$ is the lower bound of the function $f(x)$ for $x \in [x_i, x_{i+1}]$. It is also well known that the straight lines through the points $(x_{i-1}, f(x_{i-1}))$ and $(x_i, f(x_i))$ for $i = 2, \ldots, n$ and the points $(x_{i+1}, f(x_{i+1}))$ and $(x_{i+2}, f(x_{i+2}))$ for $i = 1, \ldots, n-2$ are the upper bounds of the function $f(x)$ for $x \in [x_i, x_{i+1}]$. This is illustrated in Fig. 12.

This idea can be summarized in the following lemma.

Lemma: Given n points $\{(x_i, f(x_i))|i = 1, \ldots, n\}$, where $x_1 = a < x_2 < \cdots < x_n = b$, and $f(x)$ is an concave and increasing function, the following conditions hold.

1) The piecewise linear function $\hat{f}(x) = k_i x + b_i$ if $x_i \le x \le x_{i+1}$ is the lower bound of $f(x)$, where

$$k_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}, b_i = \frac{x_{i+1} f(x_i) - x_i f(x_{i+1})}{x_{i+1} - x_i}.$$

2) The maximum gap between the piecewise linear functions $\hat{f}(x)$ and $f(x)$ is given by

$$\delta = \max_{i=1,\ldots,n-1} \delta_i \tag{27}$$

where

$$\delta_i = \begin{cases} \frac{|k_1 x_1 + b_1 - k_2 x_1 - b_2|}{\sqrt{1+k_1^2}}, & i = 1 \\ \frac{\left|\frac{k_{i-1}-k_i}{k_{i-1}-k_{i+1}}(b_{i-1}-b_{i+1}) - (b_{i-1}-b_i)\right|}{\sqrt{1+k_i^2}}, & 1 < i < n-1 \\ \frac{\|k_{n-1} x_n + b_{n-1} - k_{n-2} x_n - b_{n-2}\|}{\sqrt{1+k_{n-1}^2}}, & i = n-1. \end{cases} \tag{28}$$

Proof: The proof can easily be shown based on Fig. 12 and basic algebra geometry knowledge. We omit the proof here for space limitations.

In the following discussion, we present an iterative method for building the lower bound piecewise linear function $\hat{f}(x)$ with the predetermined approximation threshold δ. This iterative method is referred to as the sandwich algorithm in the literature [21].

The lower bound piecewise linear function and the corresponding gap are generated in an iterative way. We start evaluating the concave function $f(x)$ at the boundary points $x = a$ and $x = b$, i.e., $I = \{[a, b]\}$, $n = 2$. Then, we can obtain the piecewise linear function $\hat{f}_0(x)$ with the maximum gap of $\delta^0 = \|f(b) - f(a)\|$. Assume that, at iteration k, the maximum gap is δ^k, which is computed at the corresponding interval $[x_{j^k}, x_{j^k+1}]$. If the gap $\delta^k > \delta$, we evaluate the function $f(x)$ at the additional point $y = (x_{j^k} + x_{j^k+1})/2$. We partition the interval $[x_{j^k}, x_{j^k+1}]$ into the two intervals $[x_{j^k}, y]$ and $[y, x_{j^k+1}]$. We further evaluate the gaps for the intervals $[x_{j^k-1}, x_{j^k}]$, $[x_{j^k}, y]$, $[y, x_{j^k+1}]$, and $[x_{j^k+1}, x_{j^k+2}]$ using (27). The maximum gap is then updated. We repeat this procedure until the maximum gap is less than the given approximation threshold δ. The procedure is summarized in Algorithm 2.

This algorithm allows us to adaptively select the points $\{x_1, \ldots, x_{n_\delta}\}$ to evaluate the value of $f(x)$ based on the predetermined threshold δ. This iterative method provides us a simple way of approximating the postdecision state-value function, which is concave in the backlog x.

We have presented a method for approximating a concave function $f(x)$ with the domain of $[a, b]$ using piecewise linear functions. In this method, we can control the computation complexity and achievable performance by using different predetermined approximation error thresholds δ. The advantage of the proposed approximation method is that we can approximate the concave function only by evaluating the function at a limited number of points and without knowing the closed form of the function. We denote the aforementioned approximation operator as $A^\delta_{[a,b]} f$ for any concave function f, where the subscript $[a, b]$ emphasizes that the approximation is performed in the range of $[a, b]$. Then, $A^\delta_{[a,b]} f$ is a piecewise linear concave function and satisfies $0 \le f - A^\delta_{[a,b]} f \le \delta$. For any concave function $f(x)$ with the domain $[0, \infty)$, we approximate this concave function as follows:

$$A^\delta f = \begin{cases} A^\delta_{[0,B]} f & x \in [0, B] \\ f(B) + k^f_B (x - B) & x \in (B, \infty) \end{cases} \tag{29}$$

where k^f_B is the slope of the last segment in the piecewise linear approximation $A^\delta_{[0,B]}$, and $[0, B]$ is the range that the piecewise approximation operator is performed. In other words, we approximate the concave function $f(x)$ using the piecewise linear function (controlled by δ) in the range $[0, B]$ and using a linear function (with a slope of k) in the range of (B, ∞). It is easy to show that $A^\delta f$ is also a concave function.

Algorithm 2: Sandwich algorithm for approximating the concave function

Initialize: $x^0_1 = a$, $x^0_2 = b$, $f(x^0_1)$, $f(x^0_2)$, $\delta^0 = f(x^0_2) - f(x^0_1)$, $j^0 = 1$, $k = 0$, and $n = 2$;
Repeat:
$y = (x_{j^k} + x_{j^k+1})/2$; Compute $f(y)$;
Partition the interval $[x_{j^k}, x_{j^k+1}]$ into $[x_{j^k}, y]$ and $[y, x_{j^k+1}]$.
Compute the gaps corresponding to the intervals $[x_{j^k-1}, x_{j^k}]$, $[x_{j^k}, y]$, $[y, x_{j^k+1}]$, and $[x_{j^k+1}, x_{j^k+2}]$.
$x^{k+1}_{j+1} \leftarrow x^k_j$ for $j = j^k + 1, \ldots, n$;
$x^{k+1}_{j^k} \leftarrow y$; $x^{k+1}_j \leftarrow x^k_j$ for $j = 1, \ldots, j^k$;
$k \leftarrow k + 1$; $n \leftarrow n + 1$;
Update the maximum gap δ^k and the index j^k that correspond to the interval with the maximum gap.
Until $\delta^k \le \delta$.

Proof of Theorem 3: To prove condition 1, we need to show that $W(x, h)$ as computed in (19) is a concave function. If it is true, then $\hat{V}^{t,\lambda}(\cdot, h)$ is a piecewise linear concave function through the piecewise linear approximation defined in (18). Based on Theorem 2, we note that $J^{t,\lambda}(x, h)$ is a concave function, which shows that $W(x, h)$ is also a concave function.

To prove condition 2, we define the foresighted optimization operator as follows:

$$T_{a,h} V(x, h) = \max_{0 \le y \le x+a} \times \left[u(x + a, y) - \lambda c(h, y) + \alpha V(x + a - y, h) \right].$$

Then, the postdecision state-based Bellman equations can be rewritten as

$$V^{*,\lambda} = \mathop{\mathbf{E}}_{a,h} T_{a,h} V^{*,\lambda}$$

where $\mathbf{E}$ is the expectation over the data arrival and channel-state transition, and the operator is a maximum norm α-contraction.

The online learning of the postdecision state-value function in (17) can be re-expressed by

$$V^{t,\lambda} = V^{t-1,\lambda} + \beta_t (T_{a,h} V^{t-1,\lambda} - V^{t-1,\lambda}).$$

Similar to [20], it is shown that the convergence of the online learning algorithm is equivalent to the convergence of the following ordinary differential equation (ODE):

$$\dot{V}^\lambda = \mathop{\mathbf{E}}_{a,h} T_{a,h} V^\lambda - V^\lambda.$$

Because $T_{a,h}$ is a contraction mapping, the asymptotic stability of the unique equilibrium point of the aforementioned ODE is guaranteed [20]. This unique equilibrium point corresponds to the optimal postdecision state-value function $V^{*,\lambda}$.

When the postdecision state-value function is approximated using the approximator $A^\delta_{[0,B_t]}$, the online learning of the postdecision state-value function becomes $V^{t,\lambda} = A^\delta_{[0,B_t]}(V^{t-1,\lambda} + \beta_t (T_{a,h} V^{t-1,\lambda} - V^{t-1,\lambda}))$.

The corresponding ODE is

$$\dot{V}^{\lambda} = A^{\delta}_{[0,B_t]} \left(\mathop{\mathbf{E}}_{a,h} T_{a,h} V^{\lambda} \right) - V^{\lambda}.$$

By the contraction mapping and the property of $A^{\delta}_{[0,B_t]}$, we can show that $\|V^{*,\lambda} - \hat{V}^{\infty,\lambda}\|_{\infty} \leq \delta/(1-\alpha)$. ■

REFERENCES

[1] R. Berry and R. G. Gallager, "Communications over fading channels with delay constraints," *IEEE Trans. Inf. Theory*, vol. 48, no. 5, pp. 1135–1149, May 2002.

[2] W. Chen, M. J. Neely, and U. Mitra, "Energy-efficient transmission with individual packet delay constraints," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2090–2109, May 2008.

[3] M. Goyal, A. Kumar, and V. Sharma, "Optimal cross-layer scheduling of transmissions over a fading multiacess channel," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3518–3536, Aug. 2008.

[4] M. Agarwal, V. Borkar, and A. Karandikar, "Structural properties of optimal transmission policies over a randomly varying channel," *IEEE Trans. Autom. Control*, vol. 53, no. 6, pp. 1476–1491, Jul. 2008.

[5] D. Djonin and V. Krishnamurthy, "Structural results on optimal transmission scheduling over dynamical fading channels: A constrained Markov decision process Approach," in *Wireless Communications*, G. Yin, Ed. New York: Springer Verlag, 2006, pp. 75–98.

[6] T. Holliday, A. Goldsmith, and P. Glynn, "Optimal power control and source–channel coding for delay constrained traffic over wireless channels," in *Proc. IEEE Int. Conf. Commun.*, May 2002, vol. 2, pp. 831–835.

[7] A. Jalali, R. Padovani, and R. Pankaj, "Data throughput of CDMA-HDR: A high-efficiency data rate personal communication wireless system," in *Proc. IEEE Veh. Technol. Conf.*, May 2000, pp. 1854–1858.

[8] L. Tassiulas and A. Ephremides, "Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1949, Dec. 1992.

[9] P. R. Kumar and S. P. Meyn, "Stability of queuing networks and scheduling policies," *IEEE Trans. Autom. Control*, vol. 40, no. 2, pp. 251–260, Feb. 1995.

[10] A. Stolyar, "Maximizing queuing network utility subject to stability: Greedy primal–dual algorithm," *Queueing Syst., Theory Appl.*, vol. 50, no. 4, pp. 401–457, Aug. 2005.

[11] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, Apr. 2006.

[12] F. Fu and M. van der Schaar, "Decomposition principles and online learning in cross-layer optimization for delay-sensitive applications," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1401–1415, Mar. 2010.

[13] D. V. Djonin and V. Krishnamurthy, "Q-learning algorithms for constrained Markov decision processes with randomized monotone policies: Application to MIMO transmission control," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 2170–2181, May 2007.

[14] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An online learning algorithm for energy-efficient delay-constrained scheduling over a fading channel," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 732–742, May 2008.

[15] E. Altman, *Constrained Markov Decision Processes*. London, U.K.: Chapman & Hall, 1999.

[16] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA: Athena Scientific, 2005.

[17] Q. Zhang and S. A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *IEEE Trans. Commun.*, vol. 47, no. 11, pp. 1688–1692, Nov. 1999.

[18] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ: Prentice-Hall, 1987.

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

[20] V. S. Borkar and S. P. Meyn, "The ODE method for convergence of stochastic approximation and reinforcement learning," *SIAM J. Control Optim.*, vol. 38, no. 2, pp. 447–469, Jan. 2000.

[21] A. Siem, D. Hertog, and A. Hoffmann, "A method for approximating univariate convex functions using only function value evaluations," SSRN, 2007. [Online]. Available: http://ssrn.com/abstract=1012289

[22] A. Siem, D. Hertog, and A. Hoffmann, "Multivariate convex approximation and least-norm convex data-smoothing," in *Proc. ICCSA*, 2006, vol. 3, pp. 812–821.

[23] W. Powell, A. Ruszczynski, and H. Topaloglu, "Learning algorithms for separable approximation of discrete stochastic optimization problems," *Math. Oper. Res.*, vol. 29, no. 4, pp. 814–836, Nov. 2004.

[24] F. Fu and M. van der Schaar, "Structural solutions for dynamic scheduling in wireless multimedia transmission," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 5, pp. 727–739, May 2012.

[25] F. Fu and M. van der Schaar, "A systematic framework for dynamically optimizing multiuser video transmission," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 3, pp. 308–320, Apr. 2010.

[26] L. Kleinrock, *Queueing Systems*. New York: Wiley, 1975.

[27] F. Fu and M. van der Schaar. (2010, Mar.). "Structural-aware stochastic control for transmission scheduling," UCLA, Los Angeles, CA, Tech. Rep. [Online]. Available: http://medianetlab.ee.ucla.edu/papers/UCLATechReport_03_11_2010.pdf

[28] H. Li and K. Womer, "Stochastic resource-constrained project scheduling and its military applications," *MORS Phalanx*, Mar. 2011. [Online]. Available: www.umsl.edu/~lihait/files/PHALANX-Li%20and%20Womer.pdf

[29] J. Ma and W. B. Powell. (2010, May). "Convergence analysis of on-policy LSPI for multidimensional continuous state and action–space MDPs and extension with orthogonal polynomial approximation," Princeton Univ., Princeton, NJ, Tech. Rep. [Online]. Available: http://www.castlelab.princeton.edu/Papers/Ma%20Powell%20-%20Online%20convergence%20for%20high%20dimensional%20MDP.pdf

[30] V. B. Tadic, A. Doucet, and S. Singh, "Two-time-scale stochastic approximation for constrained stochastic optimization and constrained Markov decision problems," in *Proc. Amer. Control Conf.*, Jun. 2003, vol. 6, pp. 4736–4741.

[31] C. Watkins and P. Dayan, "Q-learning: Technical note," *Mach. Learn.*, vol. 8, no. 3/4, pp. 279–292, May 1992.

[32] E. Altman, B. Gaujal, and A. Hordijk, "Multimodularity, convexity and optimization properties," *Math. Oper. Res.*, vol. 25, no. 2, pp. 324–347, May 2000.

[33] D.P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA: Athena Scientific, 2007.

Fangwen Fu received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 2002 and 2005, respectively, and the Ph.D. degree from the University of California, Los Angeles, in 2010.

He was an Intern with the IBM T. J. Watson Research Center, Yorktown Heights, NY, and with DOCOMO USA Laboratories, Palo Alto, CA. He is currently a Media Architect with Intel Corporation, Folsom, CA. His research interests include wireless multimedia streaming, resource management for networks and systems, stochastic optimization, applied game theory, media hardware architecture, and video coding, processing, and analysis.

Dr. Fu received the Dimitris Chorafas Foundation Award in 2009. He was one of the top 12 Ph.D. students who were selected by IBM Research to participate in the 2008 Watson Emerging Leaders in Multimedia Workshop.

Mihaela van der Schaar (M'99–SM'04–F'10) received the Ph.D. degree in electrical engineering from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2001.

She is currently a Professor with the Department of Electrical Engineering, University of California, Los Angeles. She is the holder of 33 granted U.S. patents and three ISO awards for her contributions to the MPEG video compression and streaming international standardization activities. Her research interests include multiuser communication networks, multimedia communications, processing, and systems, online learning, network economics, and game theory.

Dr. van der Schaar received the National Science Foundation Faculty Early Career Development (NSF Career) Award in 2004, the Best Paper Award from the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY in 2005, the Okawa Foundation Award in 2006, the IBM Faculty Award in 2005, 2007, and 2008, and the Most Cited Paper Award from *EURASIP: Image Communications Journal* in 2006. She was an Associate Editor for the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE SIGNAL PROCESSING LETTERS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and *IEEE Signal Processing Magazine*.