Age of Information: Design and Analysis of Optimal Scheduling Algorithms

Yu-Pin Hsu*, Eytan Modiano†, and Lingjie Duan‡
*Department of Communication Engineering, National Taipei University
†Laboratory for Information and Decision Systems, Massachusetts Institute of Technology
‡Engineering Systems and Design Pillar, Singapore University of Technology and Design
yupinhsu@mail.ntpu.edu.tw, modiano@mit.edu, lingjie_duan@sutd.edu.sg

***Abstract*—*Age of information* is a newly proposed metric that captures delay *from an application layer perspective*. The age measures the amount of time that elapsed from the moment the mostly recently received update was generated until the present time. In this paper, we study an age minimization problem over a wireless broadcast network with many users, where only one user can be served at a time. We formulate a Markov decision process (MDP) to find *dynamic* transmission scheduling schemes, with the purpose of minimizing the long-run average age. While showing that an optimal scheduling algorithm for the MDP is a simple *stationary switch-type*, we propose a sequence of *finite-state approximations* for our infinite-state MDP and prove its convergence. We then propose both *optimal off-line and on-line* scheduling algorithms for the finite-approximate MDPs, depending on knowledge of time-varying arrivals.**

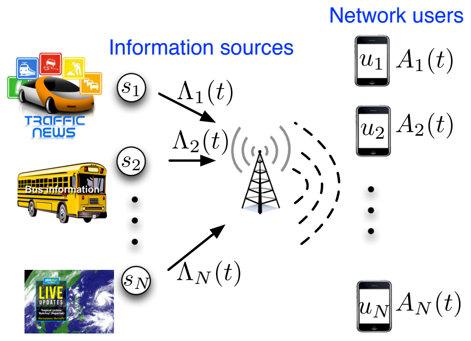

Fig. 1. A base-station updates N users $u_1, \cdots, u_N$ on information of source s_i.

I. Introduction

Ideas of traditional networks have been focused on network throughput or delay. In addition to those concerns, in recent years there has been growing interest in an *age of information*. The age is defined to capture the *freshness* of information; more precisely, it is the time elapsed since the generation of the information. This is motivated by a variety of network applications requiring *timely information*, e.g., traffic, transportation, air quality, and typhoon.

While the packet delay is usually referred to as the elapsed time from the generation to its delivery, the age includes not only the packet delay but the inter-delivery time because the age of information increases until the information is updated. We hence need to jointly consider the two parameters so as to design an age-optimal network. Moreover, while traditional relays need to keep all packets that are not served yet, the relays for the timely information only store the latest information, but remove those out-of-date packets, i.e. a new arrival always replaces the old packet in a buffer. Due to the distinctions, we need to re-consider networks for the timely information.

In resource-shared networks, the scheduling is a critical issue to optimize network performance. For delivering clean theoretical results and clear engineering insights, we look at the simplest and fundamental problem by considering a wireless broadcast network, where many network users share the common wireless medium, as shown in Fig. 1.

The work of Yu-Pin Hsu was supported by Ministry of Science and Technology, Taiwan (Project No. MOST 105-2218-E-305-003). The work of Eytan Modiano was supported by NSF award CNS-1701964. The work of Lingjie Duan was supported by Singapore Ministry of Education Academic Research Fund Tier 2 (Project No. MOE2016-T2-1-173).

A. Contribution

We consider the broadcast network in Fig. 1 associated with time-varying arrivals at the base station (BS). By formulating a Markov decision process (MDP) in Section II, we show that an optimal scheduling algorithm is *stationary* and *deterministic*; in particular, it is a simple *switch-type* algorithm, i.e., given the ages of other users, an optimal decision for a user is based on a *threshold* and the BS optimally update the user if the age of the user is larger than the threshold.

Since no practical algorithm can work on infinite-state MDPs like ours, we propose *a sequence of finite-state approximations* and rigorously show its convergence in Section IV-A. In Section IV-B, we proposed an *optimal off-line* scheduling algorithm based on the finite-state approximate MDPs. Even without knowing the arrival statistics, we also propose an *optimal on-line* scheduling algorithm in Section IV-C.

B. Related work

The *age of information* was proposed in [1]; since then, numerous works [1–7] study the age of information for a single link. The papers [1–3] consider queues to store all unserved packets (i.e., out-of-date packets are stored) and analyze the long-run average age, based on various queueing models. They show that the optimal sampling rate for minimizing the average age will not be consistent with the throughput optimum or delay optimum. The paper [4] considers a *smart* update and shows the *always update* scheme does not minimize the average age. Moreover, [5, 6] develop power-efficient updating algorithms for minimizing the average age. The model in [7] uses a small buffer to store the latest information.

Of the most relevant work to us about scheduling for multiple users is [8–10]. Both [8, 9] consider queues at a BS, while [8] develops some hueristic scheduling algorithms and [9] analyzes the complexity of scheduling for minimizing the average age. The paper [10] considers a frame-based system, where all packets arrive at a BS at the beginning of each frame, while developing an index policy. Our work is the first one to consider the age minimization problem with general arrivals, which can happen at any slot, and propose both optimal off-line and on-line scheduling algorithms.

II. System model

We consider a network consisting of a base-station (BS), and N wireless users $u_1, \cdots, u_N$ as illustrated in Fig. 1, where each user u_i is interested in the information generated by an associated source s_i. The information is transmitted through the BS in the form of packets. Our focus in this paper is on the noiseless channels, where all transmissions from the BS to each user are considered to be successful with a sufficiently high probability, i.e., we ignore transmission errors. Please see our technical report [11] for unreliable channels.

We consider a discrete-time system with slots $t = 0, 1, \cdots$. The packets from the sources (if any) arrive at the BS at the *beginning* of each slot. Without loss of generality, we assume that the BS can transmit at most one packet during each slot, i.e., the BS can update at most one user at each slot. We assume that the arrivals at the BS for different users are independent of each other and also independent and identically distributed (i.i.d.) over slots, following a Bernoulli distribution. Precisely, by $\Lambda_i(t)$, we indicate if a packet from source s_i arrives at the BS at slot t, in which $\Lambda_i(t) = 1$ if there is a packet; otherwise, $\Lambda_i(t) = 0$, where $P[\Lambda_i(t) = 1] = p_i$.

In the applications of interest in this paper, only the fresh information is significant for the users; as such, the BS needs to buffer at most one packet for each user, i.e., an arriving packet always replaces the old one in the buffer. However, in this paper we focus on the scenario without any buffer, where the arrivals are discarded if not transmitted in the current slot. The *no-buffer* system is simple for practical systems as well as results in good performance in a single link (see [7]). In our technical report [11], we have extended our results successfully by considering the buffers. It turns out that the main results hold for the *buffer-network*; moreover, the performance of the no-buffer system is very close to the buffer-network.

A. Age of information model

The *age of information* represents the freshness of the information at the *users*. We assume that the ages of the *packets* are zero when reaching the BS. On receiving a packet, the age of information for the user is reset to be one due to one slot of the transmission time.

Let $A_i(t)$ be the age of information for user u_i at slot t before the BS makes a decision. Considering a linearly increasing age over slots, the age of user u_i at slot t is $A_i(t) = t - r_i(t)$, where $r_i(t)$ is the time that the most recently received packet by user u_i was generated.

We remark that since the BS can update at most one user for each slot, $A_i(t) \geq 1$ for all i, $A_i(t) \neq A_j(t)$ for all $i \neq j$, and $\sum_{i=1}^{N} A_i(t) \geq 1 + 2 + \cdots + N$ for all t.

B. Markov decision process model

We use a Markov decision process (MDP) to develop scheduling algorithms for the BS to update a user at each transmission opportunity. According to [12], we describe the components of our MDP in detail below.

Decisions and decision epochs: For each slot, the BS makes a decision immediately after receiving packets. By $D(t) \in \{0, 1, \cdots N\}$ we denote a decision of the MDP at slot t, where $D(t) = 0$ if the BS does not transmit any packet and $D(t) = i$ for $i = 1, \cdots, N$ if user u_i is scheduled to be updated at slot t. Considering the no-buffer network, for each slot the BS updates a user with an arrival at the BS, i.e., $D(t) \in \{i : \Lambda_i(t) = 1\}$. Note that the action-space is finite.

States: We define the state $\mathbf{S}(t)$ of the system at slot t as $\mathbf{S}(t) = (A_1(t), \cdots, A_N(t), \Lambda_1(t), \cdots, \Lambda_N(t))$. By $\mathbf{S}$ we define the state-space including all possible states. Note that $\mathbf{S}$ is a *countable infinite* set because the ages are possibly unbounded.

Transition probabilities: Under decision $D(t) = d$, as the transmission time is one slot, the age of the next slot is $A_d(t+1) = 1$ and $A_i(t+1) = A_i(t) + 1$ for $i \neq d$. Let $P_{\mathbf{s},\mathbf{s}'}(d)$ be the transition probability from state $\mathbf{s} = (a_1, \cdots, a_N, \lambda_1, \cdots, \lambda_N)$ to state $\mathbf{s}' = (a'_1, \cdots, a'_N, \lambda'_1, \cdots, \lambda'_N)$ under decision $D(t) = d$, and then the probability law is

$$P_{\mathbf{s},\mathbf{s}'}(d) = \begin{cases} \Pi_{i:\lambda'_i=1} p_i \Pi_{i:\lambda'_i=0}(1-p_i) & \text{if } a'_d = 1 \text{ and} \\ & a'_i = a_i + 1 \text{ for } i \neq d; \\ 0 & \text{else.} \end{cases}$$

Cost: Let $C(\mathbf{S}(t), D(t) = d)$ be the *immediate cost* if decision $D(t) = d$ is taken at slot t under system state $\mathbf{S}(t)$. We consider the *total age* of the system after making a decision at slot t:

$$\begin{aligned} C(\mathbf{S}(t), D(t) = d) &\triangleq \sum_{i=1}^{N} A_i(t+1) \\ &= \sum_{i=1}^{N} (A_i(t) + 1) - A_d(t) \cdot \Lambda_d(t), \end{aligned}$$

where we define $A_0(t) = 0$ and $\Lambda_0(t) = 0$ for all t (for the case of $d = 0$), while the last term indicates that user u_d is updated at slot t. We remark that our analysis and design can also work perfectly with the *weighted sum* of the ages.

C. Average-optimal scheduling algorithm design

A *scheduling algorithm* θ specifies decisions at all decision epochs, i.e., $\theta = \{D(1), D(2), \cdots\}$. An algorithm is *history dependent* if $D(t)$ depends on $D(1), \cdots, D(t-1)$ and $\mathbf{S}(1) \cdots, \mathbf{S}(t)$, while it is *Markov* if $D(t)$ only depends on $\mathbf{S}(t)$. An algorithm is *stationary* if $D(t_1) = D(t_2)$

when $\mathbf{S}(t_1) = \mathbf{S}(t_2)$ for any t_1, t_2. Moreover, A *randomized* algorithm specifies a probability distribution on the set of actions for each decision epoch, while a *deterministic* algorithm chooses an action with certainty.

The *long-run average cost* for some history dependent algorithm θ is given by

$$V(\theta) = \limsup_{T\to\infty} \frac{1}{T+1} E_\theta \left[\sum_{t=0}^{T} C(\mathbf{S}(t), D(t)) | \mathbf{S}(0) \right],$$

where E_θ represents the conditional expectation, given that the policy θ is employed.

Definition 1. A scheduling algorithm θ is *average-optimal* if it minimizes the long-run average cost $V(\theta)$.

Our goal is to characterize and obtain an *average-optimal algorithm*. However, according to [12], there may not exist a stationary or deterministic algorithm that is average-optimal for an infinite state-space MDP. Due to the space limitation, we present our main results in this paper, and please see our technical report in [11] for the complete proofs.

III. Characterizations of the average-optimality

We begin with an infinite horizon α-discounted cost case, where $0 < \alpha < 1$; we then tie to the average cost case because structures of the average-optimal algorithm usually rely on the discounted cost case. Given initial state $\mathbf{S}(0) = \mathbf{s}$, the *total expected discounted cost* under an algorithm θ is

$$V_\alpha(\mathbf{s}; \theta) = \lim_{T\to\infty} E_\theta \left[\sum_{t=0}^{T} \alpha^t C(\mathbf{S}(t), D(t)) | \mathbf{S}(0) = \mathbf{s} \right].$$

Definition 2. A scheduling algorithm θ is *α-optimal* if it minimizes the total expected discounted cost $V_\alpha(\mathbf{s}; \theta)$. In particular, we define

$$V_\alpha(\mathbf{s}) = \min_\theta V_\alpha(\mathbf{s}; \theta).$$

We first introduce the *discounted cost optimality equation* for $V_\alpha(\mathbf{s})$ as below.

Proposition 3. *The optimal expected discounted cost $V_\alpha(\mathbf{s})$, for state $\mathbf{s}$, satisfies the following discounted cost optimality equation:*

$$V_\alpha(\mathbf{s}) = \min_{d\in\{0,1,\cdots,N\}} C(\mathbf{s}, d) + \alpha E[V_\alpha(\mathbf{s}')], \tag{1}$$

where the expectation is taken over all possible next state $\mathbf{s}'$ reachable from the state $\mathbf{s}$. A deterministic stationary algorithm that realizes the minimum of the right hand side (RHS) of the discounted cost optimality equation will be an α-optimal algorithm. Moreover, we define $V_{\alpha,n}(\mathbf{s})$ by $V_{\alpha,0}(\mathbf{s}) = 0$ and for any $n \geq 0$

$$V_{\alpha,n+1}(\mathbf{s}) = \min_{d\in\{0,1,\cdots,N\}} C(\mathbf{s}, d) + \alpha E[V_{\alpha,n}(\mathbf{s}')]. \tag{2}$$

Then, $V_{\alpha,n}(\mathbf{s}) \to V_\alpha(\mathbf{s})$ as $n \to \infty$ for every $\mathbf{s}$, and α.

Proof. (Sketch) Basically, we need to verify that $V_\alpha(\mathbf{s})$ is finite for all possible α and $\mathbf{s}$. Please see [11] for details. □

Lemma 4. *There exists a deterministic stationary algorithm that is average-optimal. Moreover, there exists a finite constant $g = \lim_{\alpha\to 1}(1-\alpha)V_\alpha(\mathbf{s})$ for every state $\mathbf{s}$ such that the average-optimal cost is g, independent of the initial state $\mathbf{S}(0)$.*

Proof. Please see [11]. □

We remark that even though the average-optimality of a deterministic stationary algorithm is shown in Lemma 4, we might not arrive at an average cost optimaility equation like Eq. (1) or the value iteration like Eq. (2) (see [13, 14] for details).

In addition to the average-optimality of a deterministic stationary algorithm, we show that an average-optimal scheduling algorithm has a nice structure facilitating the scheduling algorithm design in the next section.

Definition 5. A *switch-type* algorithm is a deterministic stationary algorithm: for every user u_i, if an average-optimal decision for state $\mathbf{s} = (a_1, \cdots, a_i, \cdots, a_N, \lambda_1, \cdots, \lambda_N)$ is $d^*_{\mathbf{s}} = i$, then an optimal decision at state $\mathbf{s}' = (a_1, \cdots, a_i + 1, \cdots, a_N, \lambda_1, \cdots, \lambda_N)$ is $d^*_{\mathbf{s}'} = i$ as well.

Theorem 6. *An average-optimal scheduling algorithm is the switch-type algorithm.*

Proof. (Sketch) We first prove that an α-optimal scheduling algorithm is the switch-type by applying the value iteration in Eq. (2). Then, we show that the structure holds for the average-optimum by letting $\alpha \to 1$. Please see [11] for details. □

In particular, if the arrival rates of all information sources are the same, we can obtain a nice *index algorithm* as follows.

Corollary 7. *If the arrival rates of all information sources are the same, i.e., $p_i = p_j$ for all $i \neq j$, then an optimal scheduling algorithm updates the user with the largest age, i.e., $D(t) = \arg\max_i A_i(t)$ for each time t.*

Proof. Please see [11]. □

We also notice that the index algorithm is indeed an on-line algorithm, without the knowledge of the arrival statistics in advance. For general asymmetric arrivals, an average-optimal scheduling algorithm depends on both the arrival statistics and the current ages. However, it is not obvious how to get an average-optimal scheduling for the asymmetric arrivals. This key challenge hence motivates us to investigate both off-line and on-line scheduling algorithms in the next section.

IV. Scheduling algorithm design

We start with proposing finite-state approximations to the original MDP as in practice we can only work on a finite-state MDP to avoid formidably high computation complexity. We will rigorously show the convergence of the proposed truncation as in general a MDP truncation might not converge to the original MDP according to [14].

Based on the finite-state MDPs, we first propose a structural value iteration algorithm in Section IV-B to pre-compute an optimal decision for each state, whose complexity would be lower than the conventional value iteration algorithm by employing the switch structure. Moreover, we propose an on-line algorithm leveraging the stochastic approximation [15] in Section IV-C

A. Finite-state MDP approximations

Let Δ be the Markov decision process defined in Section II-B. By $\{\Delta_m\}$ we define a sequence of approximate MDPs for Δ whose state-space is $\mathbf{S}_m = \{\mathbf{s} \in \mathbf{S} : a_i \leq m\}$ with the bounded *virtual* ages, while the decision-space and cost definition are the same as Δ.

Let $A_i^{(m)}(t)$ be the age of information for user u_i at time t for Δ_m. Different from Δ, under decision $D(t) = d$ the age of the next slot for Δ_m is $A_d^{(m)}(t+1) = 1$ and $A_i^{(m)}(t+1) = \left[A_i^{(m)}(t) + 1\right]_m^+$ if $i \neq d$, where we define the notation $[x]_m^+$ by $[x]_m^+ = x$ if $x \leq m$ and $[x]_m^+ = m$ if $x > m$.

Let $P_{\mathbf{s},\mathbf{s}'}^{(m)}(d)$ be the transition probability for Δ_m. Then, $P_{\mathbf{s},\mathbf{s}'}^{(m)}(d) = P_{\mathbf{s},\mathbf{s}'}(d)$ if $\mathrm{s}' \in \mathbf{S}_m$; otherwise,

$$P_{\mathbf{s},\mathbf{s}'}^{(m)}(d) = P_{\mathbf{s},\mathbf{s}'}(d) + \sum_{\mathbf{r} \in \mathbf{S} - \mathbf{S}_m} P_{\mathbf{s},\mathbf{r}}(d),$$

for some excess probabilities on state $\mathbf{r} \in \mathbf{S} - \mathbf{S}_m$.

Next, we show that the proposed finite-state approximations will be asymptotically average-optimal.

Theorem 8. *Let V^* and V_m^* be the average-optimal cost to Δ and Δ_m, respectively. Then, $V_m^* \to V^*$ as $m \to \infty$.*

Proof. Please see [11]. □

Now, we are ready to propose scheduling algorithms based on Δ_m. In other words, the BS make decisions according to the *virtual age* $A_i^{(m)}(t)$ on $\mathbf{S}_m$, instead of the *real age* $A_i(t)$ on $\mathbf{S}$. The real age can increase beyond m but the virtual age will be smaller than $m+1$.

B. Structural off-line scheduling algorithm

The *relative value iteration algorithm* (RVIA), as follows, can be applied to get an optimal deterministic stationary algorithm on Δ_m:

$$V_{n+1}(\mathbf{s}) = \min_{d \in \{0,1,\cdots,N\}} C(\mathbf{s}, d) + E[V_n(\mathbf{s}')] - V_n(\mathbf{0}), \quad (3)$$

for all $\mathbf{s} \in \mathbf{S}_m$, where $\mathbf{0}$ is a reference state and we can arbitrarily choose the reference state by $\mathbf{0} = (1, 2, \cdots, N, 1, \cdots, 1)$. For each iteration n, we need to update the decisions for *all* virtual states by minimizing the RHS of Eq. (3) as well as update $V(\mathbf{s})$ for *all* $\mathrm{s} \in \mathbf{S}_m$. The complexity is very high due to many users (i.e., curse of dimensionality [16]). Therefore, we propose in Alg. 1 a structural off-line algorithm based on the RVIA along with the switch structure.

In Alg. 1, we seek an optimal decision $d_{\mathbf{s}}^*$ for each virtual state $\mathrm{s} \in \mathbf{S}_m$ by iteration. For each iteration, we update both optimal decision $d_{\mathbf{s}}^*$ and $V(\mathbf{s})$ for all virtual states. If the switch property holds, we can determine an optimal decision immediately in Line 5; otherwise we find an optimal decision according to Line 7. By $V_{\text{tmp}}(\mathbf{s})$ in Line 9 we temporarily keep the updated value, which will replace $V(\mathbf{s})$ in Line 11. Using the switch structure to prevent from the minimum operations on all virtual states in the RVIA, we can reduce the computational complexity accordingly. We conclude the optimality of Alg. 1 in the following.

Algorithm 1: Structural off-line scheduling algorithm

```
1  $V(\mathbf{s}) \leftarrow 0$ for all states $\mathbf{s} \in \mathbf{S}_m$;
2  while 1 do
3      forall $\mathbf{s} = (a_1, \cdots, a_i, \cdots, a_N, \lambda_1, \cdots, \lambda_N) \in \mathbf{S}_m$ do
4          if there exists $\zeta > 0$ and $i \in \{1, \cdots, N\}$ such that $d^*_{(a_1,\cdots,a_i-\zeta,\cdots,a_N,\lambda_1,\cdots,\lambda_N)} = i$ then
5              $d_{\mathbf{s}}^* \leftarrow i$;
6          else
7              $d_{\mathbf{s}}^* \leftarrow \arg\min_{d \in \{0,1,\cdots,N\}} C(\mathbf{s}, d) + E[V(\mathbf{s}')]$;
8          end
9          $V_{\text{tmp}}(\mathbf{s}) \leftarrow C(\mathbf{s}, d_{\mathbf{s}}^*) + E[V(\mathbf{s}')] - V(\mathbf{0})$;
10     end
11     $V(\mathbf{s}) \leftarrow V_{\text{tmp}}(\mathbf{s})$ for all $\mathbf{s} \in \mathbf{S}_m$.
12 end
```

Theorem 9. *The limit point of $d_{\mathbf{s}}^*$ of Alg. 1 is an average-optimal decision for all virtual state $\mathrm{s} \in \mathbf{S}_m$. In particular, Alg. 1 converges in a finite number of iterations.*

Proof. (Sketch) We need to verify that the finite-state approximate MDPs are *unichain*. Please see [11] for details. □

C. On-line scheduling algorithm

We notice that Alg. 1 needs the arrival statistics to pre-compute an optimal decision for each virtual state. We will propose an on-line scheduling algorithm in case that the statistics is unavailable. Instead of updating $V(\mathbf{s})$ for all states at each iteration, we update $V(\mathbf{s})$ following a *sample path*, which is a set of outcomes of the arrivals over slots. It turns out that the sample-path updates will converge to the average-optimal solution.

To that end, we need a stochastic version of the RVIA. However, the RVIA in Eq. (3) is not suitable because the expectation is inside the minimization (see [16] for details). Moreover, minimizing the RHS of Eq. (3) for a given current state needs the transition probabilities to calculate the expectation. To tackle these challenges, we design *post-decision* states [16] for our problem.

We define the post-decision state $\tilde{\mathrm{s}}$ as the ages and the arrivals *after* a decision. The state we used before is referred to as the *pre-decision* state. If $\mathrm{s} = (a_1, \cdots, a_N, \lambda_1, \cdots, \lambda_N) \in \mathbf{S}_m$ is a virtual state of the system, then the virtual post-decision state after decision d is $\tilde{\mathrm{s}} = (\tilde{a}_1, \cdots, \tilde{a}_N, \tilde{\lambda}_1, \cdots, \tilde{\lambda}_N)$, where $\tilde{a}_i = 1$ if $i = d$ and $\tilde{a}_i = [a_i + 1]_m^+$ otherwise, as well as $\tilde{\lambda}_i = \lambda_i$ for all i.

Let $\tilde{V}(\tilde{\mathrm{s}})$ be a value function based on the post-decision states:

$$\tilde{V}(\tilde{\mathrm{s}}) = E_{\mathrm{s}}[V(\mathrm{s})],$$

where the expectation E_{s} is taken over all the pre-decision states reached from the post-decision state. We can then write down the post-decision average cost optimality equation [16] for the virtual post-decision age $\tilde{\mathrm{s}} = (\tilde{a}_1, \cdots, \tilde{a}_N, \tilde{\lambda}_1, \cdots, \tilde{\lambda}_N)$ with $\tilde{\mathrm{s}} \in \mathbf{S}_m$:

$$\begin{aligned}&\tilde{V}(\tilde{\mathrm{s}}) + g \\ =&E\left[\min_{d \in \{0,1,\cdots,N\}} C\left((\tilde{\mathbf{a}}, \tilde{\boldsymbol{\lambda}}'), d\right) + \tilde{V}([\tilde{\mathbf{a}} + \mathbf{1} - \tilde{\mathbf{a}}_d]_m^+, \tilde{\boldsymbol{\lambda}}')\right],\end{aligned}$$

Algorithm 2: On-line scheduling algorithm

```
   /* Initialization */
1  $\tilde{V}(\tilde{\mathbf{s}}) \leftarrow 0$ for all states $\tilde{\mathbf{s}} \in \mathbf{S}_m$;
2  $\tilde{\mathbf{s}} \leftarrow \mathbf{0}$;
3  $v \leftarrow 0$;
4  while 1 do
       /* Decision at slot $t$ */
5      We optimally make a decision $D^*(t)$ at slot $t$ according to the
         current arrivals $\mathbf{\Lambda}(t) = (\Lambda_1(t), \cdots, \Lambda_N(t)$ at slot $t$:
         $$D^*(t) = \arg\min_{d \in \{0,1,\cdots,N\}} C((\tilde{\mathbf{a}}, \mathbf{\Lambda}(t)), d) + \tilde{V}([\tilde{\mathbf{a}} + \mathbf{1} - \tilde{\mathbf{a}}_d]_m^+, \mathbf{\Lambda}(t));   (5)$$
       /* Value update */
6      $v \leftarrow C((\tilde{\mathbf{a}}, \mathbf{\Lambda}(t)), D^*(t)) + \tilde{V}([\tilde{\mathbf{a}} + \mathbf{1} - \tilde{\mathbf{a}}_{D^*(t)}]_m^+, \mathbf{\Lambda}(t)) - \tilde{V}(\mathbf{0})$;
7      $\tilde{V}(\tilde{\mathbf{s}}) \leftarrow (1 - \gamma(t))\tilde{V}(\tilde{\mathbf{s}}) + \gamma(t)v$;
       /* Post-decision state update */
8      $\tilde{\mathbf{a}} \leftarrow [\tilde{\mathbf{a}} + \mathbf{1} - \tilde{\mathbf{a}}_{D^*(t)}]_m^+$;
9      $\tilde{\boldsymbol{\lambda}} \leftarrow \mathbf{\Lambda}(t)$.
10 end
```

where $\tilde{\mathbf{a}}_i = (0, \cdots, \tilde{a}_i, \cdots, 0)$ is the zero vector except for the i-th entry being substituted by $\tilde{a}_i$, and $\mathbf{1} = (1, \cdots, 1)$ is the unit vector. Moreover, $\tilde{\boldsymbol{\lambda}}'$ is the next possible arrivals.

From above optimality equation, the RVIA is as follows:

$$\begin{aligned} &\tilde{V}_{n+1}(\tilde{\mathbf{s}}) \\ =&E\left[\min_{d \in \{0,1,\cdots,N\}} C\left((\tilde{\mathbf{a}}, \tilde{\boldsymbol{\lambda}}'), d\right) + \tilde{V}_n([\tilde{\mathbf{a}} + \mathbf{1} - \mathbf{a}_d]_m^+, \tilde{\boldsymbol{\lambda}}')\right] \\ &- \tilde{V}_n(\mathbf{0}). \end{aligned} \tag{4}$$

Then, we are ready to propose the on-line algorithm in Alg. 2 based on a stochastic version of the RVIA. In Lines 1-3, we initialize $\tilde{V}(\tilde{\mathbf{s}})$ of all virtual post-decision states and start from the reference point. Moreover, by v we record $\tilde{V}(\tilde{\mathbf{s}})$ of the current virtual post-decision state.

By observing the current arrivals $\mathbf{\Lambda}(t)$ and plugging in Eq. (4), in Line 5 we optimally update a user by minimizing Eq. (5); as such, the expectation in Eq. (4) is removed. Then, we update $\tilde{V}(\tilde{\mathbf{s}})$ of the current virtual post-decision state in Line 7, where $\gamma(t)$ is a stochastic step-size (see [16]) at slot t and hence $\tilde{V}(\tilde{\mathbf{s}})$ is balanced between the previous $\tilde{V}(\tilde{\mathbf{s}})$ and the current value v. Finally, the next virtual post-decision state is updated in Lines 8 and 9.

Theorem 10. *If $\sum_t \gamma(t) = \infty$ and $\sum_t \gamma^2(t) < \infty$, then Alg. 2 converges to the average-optimal value.*

Proof. Please see [11]. □

In the above theorem, $\sum_t \gamma(t) = \infty$ implies that Alg. 2 needs infinite number of iterations to learn the average-optimal solution, while Alg. 1 can converge to the optimal solution in a finite number of iterations. Moreover, $\sum_t \gamma^2(t) < \infty$ means that the *noise* from measuring $\tilde{V}(\tilde{\mathbf{s}})$ can be controlled.

Finally, we want to emphasize that the proposed Algs. 1 and 2 are asymptotically optimal, i.e., they converge to the optimal solutions when the finite state-space m and the slots t go to infinity. In Fig. 2, we show the performance of Algs. 1 and 2 over finite slots. In our technical report [11], we provide more numerical studies and discussions of the proposed algorithms for both no-buffer network and buffer-network.

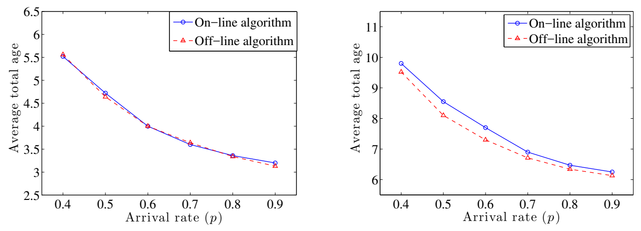

Fig. 2. The average total age over 100,000 slots versus the symmetric arrivals (i.e., $p_i = p$ for all user u_i) by employing Algs. 1 and 2 for two users (left figure) and three users (right figure).

V. Conclusion

In this paper, we consider a broadcast network, where many users are interested in different information that should be delivered by a base-station. We theoretically investigate the long-run average age of information by designing and analyzing optimal scheduling algorithms. We show that an optimal scheduling algorithm is a simple stationary switch-type. To tackle the infinite state-space Markov decision process (MDP), we propose a sequence of finite-state approximate MDPs. Based on the approximate MDPs, we propose both off-line and on-line scheduling algorithms. It turns out the proposed algorithms are asymptotically optimal.

References

[1] S. Kaul, R. D. Yates, and M. Gruteser, "Real-Time Status: How Often Should One Update?" *Proc of IEEE INFOCOM*, pp. 2731–2735, 2012.

[2] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of Message Transmission Path Diversity on Status Age," *IEEE Trans. Inf. Theory*, vol. 62, no. 3, pp. 1360–1374, 2016.

[3] L. Huang and E. Modiano, "Optimizing Age-of-Information in a Multi-Class Queueing System," *Proc. of IEEE ISIT*, pp. 1681–1685, 2015.

[4] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or Wait: How to Keep Your Data Fresh," *Proc. of IEEE INFOCOM*, pp. 1–9, 2016.

[5] R. D. Yates, "Lazy is Timely: Status Updates by an Energy Harvesting Source," *Proc. of IEEE ISIT*, pp. 3008–3012, 2015.

[6] B. T. Bacinoglu, E. T. Ceran, and E. Uysal-Biyikoglu, "Age of Information under Energy Replenishment Constraints," *Proc. of ITA*, pp. 25–31, 2015.

[7] M. Costa, M. Codreanu, and A. Ephremides, "On the Age of Information in Status Update Systems with Packet Management," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1897–1910, 2016.

[8] L. Golab, T. Johnson, and V. Shkapenyuk, "Scalable Scheduling of Updates in Streaming Data Warehouses," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 1092–1105, 2012.

[9] Q. He, D. Yuan, and A. Ephremides, "Optimizing Freshness of Information: On Minimum Age Link Scheduling in Wireless Systems," *Proc. of WiOpt*, pp. 1–8, 2016.

[10] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing Age of Information in Broadcast Wireless Networks," *Proc. of Allerton*, 2016.

[11] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of Information: Design and Analysis of Optimal Scheduling Algorithms," *Technical report*, 2017. [Online]. Available: http://web.ntpu.edu.tw/~yupinhsu/age.pdf

[12] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. The MIT Press, 1994.

[13] L. I. Sennott, "Average Cost Optimal Stationary Policies in Infinite State Markov Decision Processes with Unbounded Costs," *Operations Research*, vol. 37, pp. 626–633, 1989.

[14] ——, *Stochastic Dynamic Programming and the Control of Queueing Systems*. John Wiley & Sons, 1998.

[15] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[16] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2011.