An Online Learning Approach to Optimizing Time-Varying Costs of AoI

Vishrant Tripathi
vishrant@mit.edu
Massachusetts Institute of Technology
USA

Eytan Modiano
modiano@mit.edu
Massachusetts Institute of Technology
USA

ABSTRACT

We consider systems that require timely monitoring of sources over a communication network, where the cost of delayed information is unknown, time-varying and possibly adversarial. For the single source monitoring problem, we design algorithms that achieve sublinear regret compared to the best fixed policy in hindsight. For the multiple source scheduling problem, we design a new online learning algorithm called *Follow the Perturbed Whittle Leader* and show that it has low regret compared to the best fixed scheduling policy in hindsight, while remaining computationally feasible. The algorithm and its regret analysis are novel and of independent interest to the study of online restless multi-armed bandit problems. We further design algorithms that achieve sublinear regret compared to the best dynamic policy when the environment is slowly varying. Finally, we apply our algorithms to a mobility tracking problem. We consider non-stationary and adversarial mobility models and illustrate the performance benefit of using our online learning algorithms compared to an oblivious scheduling policy.

CCS CONCEPTS

• **Networks → Network performance modeling**; **Network performance analysis**; **Mobile ad hoc networks**.

KEYWORDS

Age of Information, wireless networks, online learning, scheduling

ACM Reference Format:
Vishrant Tripathi and Eytan Modiano. 2021. An Online Learning Approach to Optimizing Time-Varying Costs of AoI. In *The Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21), July 26–29, 2021, Shanghai, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3466772.3467053

1 INTRODUCTION

Monitoring, estimation, and control of systems are fundamental and well studied problems. Many emerging applications involve performing these tasks over communication networks. Examples include: networked control systems, sensing for IoT applications, control of robot swarms, real-time surveillance, and monitoring of sensor networks. In these settings, achieving good performance requires timely delivery of status updates from sources to destinations.

Age of Information (AoI) is a metric that captures timeliness of received information at a destination [21, 23, 33]. Unlike packet delay, AoI measures the lag in obtaining information at a destination node, and is therefore suited for applications involving time sensitive updates. Age of information, at a destination, is defined as the time that has elapsed since the last received information update was generated at the source. AoI, upon reception of a new update, drops to the time elapsed since generation of the update, and grows linearly otherwise. Over the past few years, there has been a rapidly growing body of work on analyzing AoI for queuing systems [3, 14, 15, 21, 23, 33], using AoI as a metric for scheduling policies in networks [11, 18, 19, 34, 35, 37] and for monitoring or controlling systems over networks [8, 24, 28, 30, 32]. For detailed surveys of AoI literature see [25] and [31].

Typically, AoI is used as a metric for measuring freshness of information being delivered about a source to a monitoring station. It represents a measure of distortion between the state of the system that is expected at the monitor based on past updates and the actual current state of the system. Thus, a larger age corresponds to the monitor having a higher uncertainty about the current state of the system being observed. This, in turn, means that ensuring a low average AoI can lead to higher monitoring accuracy or better control performance. While AoI is a proxy for measuring the cost of having out-of-date information, it may not properly reflect the impact of stale information on system performance.

When multiple systems or sources are being observed at the same time, there arises a need to differentiate between them based on their relative importance. Many works on AoI-based scheduling for multiple sources consider weighted-sum AoI minimization [18, 19, 34], where weights represent the relative importance of each source. Typical assumptions involve the weights being fixed and known in advance, based on the underlying application or systems being monitored.

Further, recent works on networked control systems [8, 24] and remote estimation [28, 30, 32] emphasize that even for very simple systems, linear AoI is not a sufficiently accurate metric to track accuracy or overall system performance. This has motivated interest in using general, possibly non-linear cost functions of AoI that reflect the cost of delayed information more accurately [8, 17, 26, 35]. Typical assumptions in works studying non-linear AoI include knowing the cost functions in advance [17, 24, 25, 35], assuming that cost functions increase monotonically with AoI [24, 25, 30, 37] and decoupled costs across multiple systems [24, 35]. Learning how to sample a source through a network with an unknown delay profile while minimizing AoI has been considered in [22]. Minimizing AoI with unknown and adversarial channel processes has also been considered in [6] and [2], respectively.

Importantly, we observe that all of these works assume there is some *fixed and known* cost function mapping the AoI to the cost of delayed information. In this work, we ask the question: *what if this cost function is not known in advance, time-varying and possibly adversarial?* How does one go about designing scheduling policies that lead to good monitoring accuracy or control? Related to our work, a context-aware notion of AoI was proposed in [39], where the authors considered sources with *known* time-varying context that influences the AoI cost function.

Our goal is to model applications where delivering timely information is of essence but the costs for delayed information are not completely known beforehand and hard to model, including non-stationary settings and adversarial dynamics. A broad range of networked control and monitoring applications fit this description. An example is designing scheduling schemes for real-time monitoring of power grids which have nonlinear and complicated dynamics that cannot be easily modeled. Another example is scheduling for mobility tracking. Mobility traces in the real world are often highly non-stationary and hard to explain via models. A third example is monitoring queue length information in data centers for load balancing, where only a small number of queues are sampled every few time-steps, and traffic flows, server outages and job sizes may be non-stationary and possibly adversarial. All of the above problems require optimization of unknown time-varying cost functions of AoI in an online fashion.

In Section 2 we formulate a problem that involves monitoring a single non-stationary source over a costly communication channel. We design an epoch based framework in which the AoI cost functions change across epochs in an unknown time-varying manner, but remain fixed within an epoch. At the end of each epoch, the scheduler receives feedback (either partial or full) regarding the cost in the previous epoch and uses it to decide a policy for the next epoch. We provide simple scheduling algorithms that have sublinear worst-case regret compared to the best fixed policy in hindsight. Our main contribution here is formulating the problem in such a way that we can apply techniques from online optimization. To the best of our knowledge, this is the first work to study monitoring and scheduling for non-stationary sources with unknown dynamics.

In Section 3, we use insights from the singe source model to develop an epoch based framework for online scheduling of multiple sources. In each epoch, the scheduler needs to decide a scheduling policy that specifies which source gets to send an update in every time-slot. The goal is to dynamically adapt the scheduling policy to optimize for overall monitoring cost, as the AoI cost functions change across epochs in an unknown manner. Since the number of scheduling policies grows exponentially in the length of the time horizon, it becomes computationally infeasible to implement traditional online learning algorithms directly in the multiple source setting. We design a new online learning algorithm called *Follow the Perturbed Whittle Leader* (FPWL) for this setting that is computationally feasible while also achieving low regret. Here, analyzing regret is especially challenging due to the combinatorial nature of the scheduling problem and since the Whittle index is only an approximately optimal solution for the offline scheduling problem. Our algorithm and its regret analysis are novel and of independent interest to the study of online learning for restless multi-armed bandits with time-varying costs. We also note that due to the epoch structure of our framework, it is better suited to settings when the AoI cost functions don't change rapidly (in every time-slot) but over longer time intervals.

In Section 4, we apply the algorithms that we develop to a mobility tracking problem and illustrate the performance benefits of using online learning for scheduling.

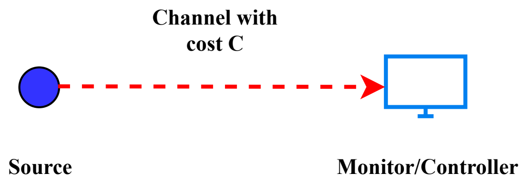

Figure 1: Single source monitoring

2 SINGLE SOURCE MONITORING

We start by discussing the single source setting with a known AoI cost function that remains fixed throughout. This will provide important insight and reveal key technical issues while formulating an online version of the problem.

Consider a single source sending updates to a monitoring station over a costly wireless channel as in Figure 1. In every time-slot, the scheduler decides whether the source sends a new update to the monitor. If it does, the monitor receives a new update in the next time-slot. The monitor maintains an age of information $A(t)$ which tracks how long it has been since it received a new update from the source. The evolution of $A(t)$ can be written as:

$$A(t+1) = \begin{cases} A(t)+1, & \text{if } u(t)=0 \\ 1, & \text{if } u(t)=1, \end{cases} \tag{1}$$

where $u(t)$ indicates whether a new update was sent in time-slot t. The function $f(\cdot)$ models the cost of having stale information at the monitor. Thus, the cost at any time-slot is:

$$\text{Cost}(t) = f(A(t)) + Cu(t), \tag{2}$$

where $C > 0$ is the cost of sending a new update from the source. Our goal is to design a monitoring policy that minimizes the long term average cost. The average cost under a policy π is:

$$\text{Cost}_{\text{ave.}}(\pi) \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} f(A^{\pi}(t)) + Cu^{\pi}(t). \tag{3}$$

This problem and the optimal policy have been analyzed in [35] and [24]. We describe the key result from these works below.

Lemma 1. *The optimal policy for the single source monitoring problem is a stationary threshold policy. Let H satisfy*

$$f(H) \leq \frac{\sum_{j=1}^{H} f(j) + C}{H} \leq f(H+1). \tag{4}$$

Then, the optimal policy is to send an update at time-slot t only if $A(t) \geq H$. If no such H exists, the optimal policy is to never send an update.

Proof. See Theorem 1 in [35]. □

Lemma 1 implies that the optimal monitoring policy is to send an update only if the current AoI gets above a threshold H. Similar threshold based schemes have appeared in many different settings for online sampling and remote estimation [28, 30, 32, 38].

Now consider a naive reformulation of the problem where the cost function $f(\cdot)$ is time-varying and unknown. We represent it by $f_t(\cdot)$ where the subscript indicates its time varying nature. If the function were fixed and unknown, we could have used reinforcement learning to solve the problem as done in [22]. However, this cannot be done for settings with time-varying costs.

On the other hand, there is a large amount of literature on online learning and optimization where the goal is to solve a sequence of optimization problems which vary in an unknown, possibly adversarial manner (see [7] and [13] for a detailed introduction to the field). In these problems there is no system state or history, so decisions in the current time-slot do not affect the optimization problem or decisions in a future time-slot. This is not true of our monitoring problem which has a system state (AoI), and where the state evolution depends on decisions taken in the past. To overcome these difficulties, we reformulate the single source monitoring problem in an epoch based setting.

2.1 An Epoch Based Formulation

We observe that the AoI of the source resets to 1 after every new update delivery. Thus, AoI evolution within an update inter-delivery period does not depend on the AoI evolution in any other period. We use this observation to formulate an epoch based problem.

We divide time into T epochs, where each epoch further consists of M time-slots. As before, when a new update is sent it gets delivered in the next time-slot. At the beginning of epoch k, we choose an AoI threshold $x_k \in \{1, ..., M\}$. Within the epoch, the source generates a new sample and sends it to the monitor whenever the AoI reaches the threshold x_k. In the last time-slot of the epoch, a new sample is sent regardless of the AoI. This ensures that the next epoch begins with an AoI of 1. A cost is observed for sending samples every x_k time-slots based on the current system dynamics and communication costs. Then, epoch $k+1$ starts. Using cost information about previous decisions, a new sampling threshold x_{k+1} is chosen for epoch $k+1$ and the process repeats itself.

In each epoch k there is a function $f_k(\cdot)$ that represents the current cost for age of information and remains fixed for the duration of the epoch. So, for any time-slot t in epoch k, the current cost is given by $f_k(A(t)) + Cu(t)$. The total cost incurred in epoch k denoted by $C_k(x)$ is simply the sum of the cost in the individual time-slots.

Lemma 2. *If a sampling threshold of x is chosen in epoch k and the AoI cost function is $f_k(\cdot)$ then the cost $C_k(x)$ is given by:*

$$C_k(x) = \left\lfloor \frac{M}{x} \right\rfloor \left(\sum_{j=1}^{x} f_k(j) + C \right) + \mathbb{1}_{r>0} \left(\sum_{j=1}^{r} f_k(j) + C \right), \tag{5}$$

where $r = M \mod x$. This is the sum total AoI cost of monitoring over the epoch k.

Proof. See Appendix A. □

If we knew $f_k(\cdot)$ at the beginning of epoch k, we could use (5) to find the optimal sampling threshold x_k^*. In our online framework, the goal is to learn the best sampling thresholds without knowing any information about the sequence of cost functions that we are going to face.

While we have motivated the setting above using cost that splits into a sum of AoI cost and communication cost, our setup allows for general cost functions $C_k(x_k)$ that map the choice of sampling threshold x_k to a cost in epoch k. For the remainder of this section, we will deal with these general cost functions $C_k(\cdot)$.

In our online setting, an **unconstrained adversary** chooses the sequence of **bounded** cost functions $C_k(\cdot)$ for each epoch k. The designer does not have access to the sequence of cost functions beforehand and must learn a suitable transmission/sampling policy in an online manner. We make no assumptions on how the underlying system dynamics or resulting cost functions change across epochs.

Note that the cost function $C_k(\cdot)$ in epoch k can be seen as an M dimensional vector where the cost for choosing the sampling threshold x is represented by the $C_k(x)$ which is the xth element of the vector. Going forward, when we use the notation C_k, we refer to the M dimensional vector of costs for each threshold in epoch t, while $C_k(x)$ represents its xth element.

The boundedness of the cost functions C_k is crucial to proving any meaningful results in this setting and is standard in online learning literature. Without loss of generality, we further assume that the cost functions are normalized such that $C_k(x) \in [0, 1]$ for all sampling thresholds x and epochs k.

2.1.1 *Feedback Structure.*

For the setup described above, we will look at two kinds of feedback structure for observing the costs. Note that x_k represents the decision made at the beginning of epoch k.

- Full Feedback - the scheduler observes the entire cost function $C_k(x), \forall x \in \{1, ..., M\}$ at the end of epoch k.
- Bandit Feedback - the scheduler observes only $C_k(x_k)$ at the end of epoch k.

2.1.2 *Objective (Regret Minimization).*

For any sequence of cost functions $C_1(\cdot), C_2(\cdot), ..., C_T(\cdot)$, x^* is defined as the best fixed sampling threshold that minimizes sum AoI cost, i.e.

$$x^* \triangleq \underset{x \in \{1,...,M\}}{\arg\min} \sum_{k=1}^{T} C_k(x). \tag{6}$$

Our goal is to find an online policy that achieves sublinear regret compared to the best fixed sampling threshold x^* for any sequence. This is known as *worst-case static regret.* For any policy π, it is defined as follows:

$$\text{Regret}_T(\pi) = \sup_{C_1,...,C_T} \left\{ \sum_{k=1}^{T} C_k(x_k^{\pi}) - \min_{x \in \{1,...,M\}} \sum_{k=1}^{T} C_k(x) \right\}. \tag{7}$$

Note that regret is defined over epochs rather than time-slots since we assume that cost functions can change only across epochs.

We will now show that our online sampling problem formulation is equivalent to the prediction with expert advice setting that is well studied in online learning literature. This will allow us to apply policies and regret bounds derived for this setting to our problem.

2.1.3 *Prediction With Expert Advice:*

A decision maker has to choose among the advice of n given experts. After making a choice, a bounded loss is incurred. This scenario is repeated iteratively, and at each iteration the costs of choosing the various experts are arbitrary (possibly even adversarial, trying to mislead the decision maker). The goal of the decision maker is to do as well as the best expert in hindsight.

In our setting, the role of experts is played by the AoI thresholds $x \in \{1, ..., M\}$. In each epoch, the scheduler decides an AoI sampling threshold x and observes an associated cost. This process repeats iteratively with time-varying, possibly adversarial changes to costs. Thus, our setting corresponds with the expert advice setting with M experts.

2.1.4 Sublinear Regret. We now discuss in detail a policy that achieves sublinear static regret for the full feedback setting. This will illustrate how regret bounds from the online learning literature can be applied to our single source online monitoring setup.

The full feedback assumption in our setting means that we observe costs for all possible sampling thresholds in every epoch. This makes sense when the scheduler has information about the source dynamics and communication costs at the end of every epoch. Knowing this information is often sufficient to construct the current cost function for any possible sampling threshold.

We describe an online monitoring policy based on Follow the Perturbed Leader (FTPL) style algorithms. The FTPL method was first analyzed in the online setting in [20] and is based on an algorithm first proposed in [12]. The key idea of the FTPL algorithm is to maintain the sum of cost functions observed until now and perturb it slightly. Choosing the best AoI threshold based on this perturbed history is sufficient to get sublinear regret.

Algorithm 1: FTPL for Online Monitoring

```
Input :parameter $\eta > 0$, number of thresholds $M$
1 Set $\Theta_1 \leftarrow 0$
2 while $t \in 1, ..., T$ do
3     Sample $\gamma_t \sim \mathcal{N}(0, I)$
4     Choose sampling threshold
          $x_t \in \arg\min_{x \in \{1,...,M\}} \Theta_t(x) + \eta\gamma_t(x)$
5     Incur loss $C_t(x_t)$ and update $\Theta_{t+1} = \Theta_t + C_t$
6 end
```

Theorem 1. *FTPL online monitoring described by Algorithm 1 with $\eta = \sqrt{T}$ achieves the following upper bound for expected regret:*

$$\mathbb{E}[Regret_T(FTPL)] \leq 2\sqrt{2T \log M},$$

where the expectation is taken over the random perturbations.

Proof. The proof is based on [10]. A lower bound of the form $\Omega(\sqrt{T \log M})$ is also available in [7]. □

Just as for the full feedback case, algorithms that achieve sublinear static regret in the bandit feedback case have been well studied for prediction with experts (see EXP3 proposed in [1]). We can directly apply them to our online monitoring formulation to design sampling policies that achieve sublinear regret. We include a review of these methods and their application to our setting in our technical report [36].

Next, we discuss what sublinear epoch regret means for AoI cost averaged over time-slots. To do so, we note that the sum total AoI cost over all time-slots equals the cost summed over individual epochs, by definition. Let π denote an online algorithm which specifies threshold x_k^{π} to be chosen in epoch k and let E_k be the set of times-slots in epoch k. Then,

$$\sum_{k=1}^{T} C_k(x_k^{\pi}) = \sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^{\pi}(t)) + Cu^{\pi}(t). \tag{8}$$

The relation above immediately implies that epoch regret also equals regret over time-slots. If π has an upper bound on its expected static epoch regret of the form $R(M, T)$ and π^* denotes the policy corresponding to the best fixed AoI threshold x^*, then substituting $\sum_{k=1}^{T} C_k(x_k^{\pi})$ in (7) using (8) we get:

$$\mathbb{E}\Bigg[\sup_{f_1,...,f_T} \Bigg\{\sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^{\pi}(t)) + Cu^{\pi}(t) - \sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^{\pi^*}(t)) + Cu^{\pi^*}(t)\Bigg\}\Bigg] \leq R(M, T). \tag{9}$$

The LHS of the equation above is simply the regret defined over time-slots rather than epochs. If $R(M, T)$ is sublinear in the number of epochs T, then it is also sublinear in the number of time-slots MT since we assume that M is fixed to be a large constant. Thus, sublinear epoch regret implies sublinear time-slot regret. As a direct corollary of this, any online algorithm with sublinear static epoch regret achieves an expected time-average AoI cost which is at least as good as that under the best fixed sampling threshold.

Corollary 1. *Suppose an online algorithm π has an upper bound on its expected static regret that grows sublinearly in T. Let π^* denote the policy corresponding to the best fixed AoI threshold x^*. Then for any sequence of bounded cost functions*

$$\limsup_{T \to \infty} \frac{1}{MT} \mathbb{E}\Bigg[\sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^{\pi}(t)) + Cu^{\pi}(t)\Bigg] \leq \limsup_{T \to \infty} \frac{1}{MT} \mathbb{E}\Bigg[\sum_{k=1}^{T} \sum_{t \in E_k} f_k(A^{\pi^*}(t)) + Cu^{\pi^*}(t)\Bigg]. \tag{10}$$

Proof. See Appendix B in the technical report [36]. □

Thus, in a time-average sense, it is sufficient to design monitoring policies with sublinear epoch regret and they are guaranteed to work as well as the best fixed threshold. Note that the relation above is an inequality and not an equality because we are comparing to the best static threshold policy across epochs and it is possible that our online monitoring policy performs better.

3 MULTIPLE SOURCES

Motivated by the single-source discussion, we study the more challenging problem, where multiple sources are sending information to a monitoring station over a network as in Figure 2. In this setting, the scheduler needs to decide which source gets to send an update in every time-slot to optimize for overall monitoring accuracy and performance, and the goal is to learn good scheduling policies.

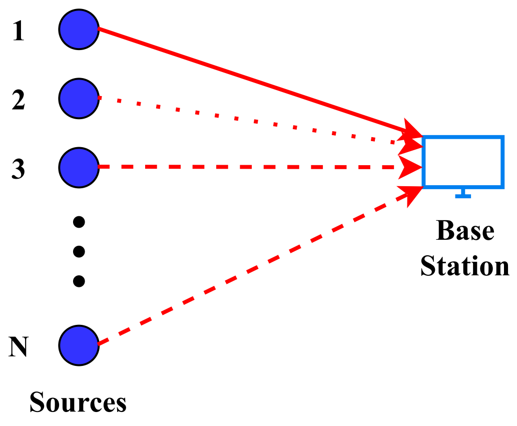

Figure 2: Multiple source monitoring

Consider a system with N sources sending updates over a network such that only one source can transmit at any given time-slot (due to interference/capacity constraints). We assume reliable channels, i.e. when a source is chosen to transmit an update, it is delivered to the monitor without fail in the next time-slot. Freshness aware scheduling in such single-hop wireless networks has been the focus of a lot of recent work in the AoI community [11, 17–19, 34, 35, 37].

We now create an epoch-based structure and set up an online learning formulation for multiple sources as we did in the single source setting. As before, we divide the time horizon into T epochs, where each epoch is of length M time-slots. At the beginning of epoch k, the scheduler needs to decide a scheduling policy π_k which specifies when to schedule each sensor. Once the epoch is over, a cost of the form $C_k(\pi_k)$ is incurred and a new epoch begins. Using cost information about previous decisions, we again choose a scheduling policy π_{k+1} for epoch $k+1$ and the process repeats itself.

We maintain variables $A^{(1)}, ..., A^{(N)}$ which track the evolution of AoI for each source within an epoch. The evolution of AoI for source i in epoch k is described by the following equation:

$$A^{(i)}(j+1) = \begin{cases} A^{(i)}(j)+1, & \text{if } i \notin \pi_k(j) \\ 1, & \text{if } i \in \pi_k(j), \end{cases} \tag{11}$$

where j is an index denoting the current time-slot within the epoch and $\pi_k(j)$ is the scheduling decision set in time-slot j of epoch k.

Similar to the single-source formulation, we relax the interference constraint in the last time-slot of every epoch. This ensures that the AoI of every source is set to 1 at the beginning of each epoch and we do not need to maintain history of AoI across different epochs. Practically, we justify this as a two time-scale assumption. A scheduling policy remains fixed over an epoch (the longer time-scale) and specifies how to make decisions over time-slots (the shorter time-scale). Once the epoch ends, the system resets. The system designer observes the performance of the scheduling policy that was chosen and specifies a new scheduling policy for the next epoch.

We consider scheduling policies as a sequence of M scheduling decisions, specifying which source gets to transmit in each time-slot within an epoch. We denote this space of scheduling policies by Π^M.

We assume that the cost for delayed information in any time-slot can be represented as a general function of the current AoIs. Let $f_k(A^{(1)}, ..., A^{(N)})$ represent this AoI cost function in epoch k, where $f_k : \mathbb{Z}^{+N} \to [0, D]$ is a **bounded** mapping from the set of AoI vectors to costs. The total cost of choosing a policy π in epoch k is given by

$$C_k(\pi) = \sum_{j=1}^{M} f_k(A^{(1)}(j), ..., A^{(N)}(j)), \tag{12}$$

where the AoIs evolve under policy π according to (11).

We have an **unconstrained adversary** who chooses the sequence of bounded cost functions $f_k(\cdot)$ for each epoch k and we need to learn the best scheduling policy in response to any sequence of cost functions. Without loss of generality we assume that $f_k(\cdot)$ are normalized such that the total cost of any policy $C_k(\cdot)$, given by (12), lies in the set $[0, 1]$.

At the end of every epoch, the scheduler receives feedback in terms of $C_k(\cdot)$. In the case of full feedback, the entire function $C_k(\cdot)$ is revealed, meaning cost for all scheduling policies is known when the epoch ends. For the case of bandit feedback, only the cost for the chosen scheduling policy $C_k(\pi_k)$ is revealed.

Observe that the multiple source problem with the feedback structure as above can also be viewed as prediction with expert advice. Now, instead of AoI thresholds as experts, we have scheduling policies as experts and our goal is to *compete with the best scheduling policy in hindsight.*

Thus, we can directly apply online learning algorithms as done in Section 2 to the multiple source setting. The regret bounds, however, are not the same. This is because the number of experts, i.e. scheduling policies of length M time-slots, scales as $\Theta(N^M)$.

Lemma 3. *Consider the multiple source online scheduling problem with $N \geq 2$. If an online algorithm has an upper bound $R(M, T)$ on its expected regret in the single source setting, then the same algorithm run using scheduling policies as experts for the multiple source problem has the following regret bound:*

$$\mathbb{E}[Regret_T(Alg.)] \leq \bar{C}R(N^M, T),$$

where $\bar{C} > 0$ is a constant that does not depend on any other parameters.

We observe that while the dependence of regret on T remains the same, it becomes exponentially worse in M for the multiple source setting. This also highlights a key computational challenge in the multiple source setting. The number of policies scales exponentially with M, the length of an epoch. Thus the optimization step in FTPL (Algorithm 1) has computational complexity that scales exponentially with M. Similar computational challenges are faced in implementing exponential weight algorithms like EXP3 for the bandit feedback case of the multiple source setting. This makes it hard to implement these online scheduling schemes in practice.

This is not surprising, given that the offline problem of finding the best scheduling policy of length M time-slots in the setting with cost functions known beforehand also requires computation that scales as $O(N^M)$ (see [17]). In [35], the authors analyzed the setting where cost functions can be represented as sums of separate cost functions that depend only on the AoI of each source individually. If the individual cost functions of AoI for each source are monotone increasing, then a low complexity heuristic based on the Whittle index approach can be found which is nearly optimal. We will use this observation to design low complexity online policies that keep track of the best scheduling policy in hindsight.

3.1 Online Whittle-Index Scheduling

We modify the general multiple source setting so as to solve the computational challenge discussed above. First, we assume that the cost function can be represented as a sum of N separate cost functions of AoI, where $f_k^{(1)},...,f_k^{(N)}$ represent individual AoI cost functions for each source in epoch k. Then, the total cost of choosing a policy π in epoch k is given by

$$C_k(\pi) = \frac{1}{NM}\sum_{j=1}^{M}\sum_{i=1}^{N} f_k^{(i)}(A^{(i)}(j)), \tag{13}$$

where the AoIs evolve under policy π according to (11). We include a normalizing constant $\frac{1}{NM}$ to the sum AoI cost to make regret analysis cleaner.

We assume that the cost functions $f_k^{(i)} : \mathbb{Z}^+ \to \mathbb{R}^+$ are fixed during an epoch and bounded monotone increasing functions of AoI, i.e. if $x > y$ then $f_k^{(i)}(x) \geq f_k^{(i)}(y)$ and $f_k^{(i)}(\cdot) \leq D$. An unconstrained adversary is free to change these bounded cost functions arbitrarily across epochs.

Second, instead of receiving feedback directly in terms of cost of scheduling policies $C_k : \Pi^M \to [0,1]$, we consider feedback in terms of individual cost functions of AoI. So, at the end of epoch k, a cost $C_k(\pi)$ is incurred (given by (13)) and AoI cost functions $f_k^{(1)},...,f_k^{(N)}$ are revealed to the scheduler, either completely or partially. In the case of bandit feedback, we will construct estimates of the entire cost functions $\hat{f}_k^{(1)},...,\hat{f}_k^{(N)}$, using linear interpolation.

Note that within an epoch, the scheduling problem that we want to solve is an instantiation of the functions of age problem described in [35]. We briefly review the multiple source setting studied in [35] below.

Consider N sources and a given set of increasing AoI cost functions $f^{(1)},...,f^{(N)}$. Our goal is to minimize average age cost over an infinite horizon. The Whittle index policy maps the current vector of source AoIs to a scheduling decision. If the current AoI for source i is $A^{(i)}$ then the Whittle policy is given by

$$\pi^{\text{Whittle}}(A^{(1)},...,A^{(N)}) \triangleq \arg\max_{i\in\{1,...,N\}} \{W^{(i)}(A^{(i)})\}, \tag{14}$$

where

$$W^{(i)}(x) \triangleq xf^{(i)}(x+1) - \sum_{k=1}^{x} f^{(i)}(k)$$

are Whittle index functions. It was shown in [35] that this Whittle policy is optimal for $N = 2$ and near optimal in general. For cost functions $f^{(1)},...,f^{(N)}$, we denote the Whittle policy given by (14) as $\text{Whittle}\big(f^{(1)},...,f^{(N)}\big)$. Next, we describe how to design a low-complexity online algorithm using Whittle index policies.

3.1.1 *Full Feedback.*

In this setting, we assume that the entire M dimensional AoI cost function $f_k^{(i)}$ for each source i is revealed to the scheduler at the end of the epoch. Instead of looking for the best schedule in every epoch which is computationally expensive, we will use the Whittle index policy as an approximate minimizer. This leads to Algorithm 2, which we call *Follow the Perturbed Whittle Leader* (FPWL).

Algorithm 2: Follow the Perturbed Whittle Leader

```
Input :parameter $\epsilon > 0$
1 Set $F_1^{(i)}(j) = j, \forall i \in \{1,...,N\}, \forall j \in \{1,...,M\}$
2 while $t \in 1,...,T$ do
3     Set $A^{(1)},...,A^{(N)} = \mathbf{1}$
4     Sample $\delta_t^{(i)}(j) \sim$ uniform in $[0, 1/\epsilon]$, i.i.d. $\forall i \in$
        $\{1,...,N\}$ and $\forall j \in \{1,...,M\}$
5     Compute $\gamma_t^{(i)}(j) = \sum_{k=1}^{j} \delta_t^{(i)}(k), \forall i, j$
6     Choose scheduling policy
        $\pi_t = \text{Whittle}\left(F_t^{(1)} + \gamma_t^{(1)},...,F_t^{(N)} + \gamma_t^{(N)}\right)$
7     Incur loss = $C_t(\pi_t)$ over epoch $t$ and observe feedback
        on $f_t^{(1)},...,f_t^{(N)}$
8     In case of bandit feedback, construct cost estimates
        $\hat{f}_t^{(i)}, \forall i \in \{1,...,N\}$ using linear interpolation
9     Update
        $F_{t+1}^{(i)} = \begin{cases} F_t^{(i)} + f_t^{(i)}, \forall i \in \{1,...,N\}, & \text{if full feedback} \\ F_t^{(i)} + \hat{f}_t^{(i)}, \forall i \in \{1,...,N\}, & \text{if bandit feedback.} \end{cases}$
10 end
```

FPWL can be divided into three major steps. First, accumulate the entire history of cost functions that the scheduler has seen until the current epoch in $F_t^{(1)},...,F_t^{(N)}$ (step 9 in Algorithm 2). Since cost functions in each epoch are increasing in terms of AoI, their sums $F_t^{(i)}$ are also increasing. Second, perturb these accumulated cost functions in a manner such that they remain increasing functions of AoI but are still amenable for FTPL style analysis (steps 4 and 5 in Algorithm 2). Third, instead of computing the best possible scheduling policy for these accumulated and perturbed cost functions, use the Whittle index policy as an approximate minimizer (step 6 in Algorithm 2).

Importantly, we require that our perturbations to the accumulated cost functions maintain their monotonicity. This is because the Whittle index policy from [35] is known to be a good approximate solution only if the cost functions are increasing. Simply adding i.i.d. uniform random vectors, as is typical in follow-the-leader algorithms is not sufficient to ensure that cost functions remain increasing. This is why we first create uniform random vectors $\delta_t^{(i)}$ in step 4 and then construct random and monotone perturbations $\gamma_t^{(i)}$ for each cost function in step 5.

Computing the Whittle policy has complexity $O(NM)$ since it involves a maximization over N quantities for at most M steps. Further, generating the random perturbations γ_t in steps 4 and 5 also takes at most $O(NM)$ computation. Thus, the algorithm above resolves the computational challenge involved in implementing FTPL directly for the online scheduling problem.

Proving regret bounds for our proposed algorithm is much harder than in the settings studied earlier. We overcome three significant problems: 1) perturbations in Algorithm 2 are made to the AoI cost functions rather than policies, unlike regular FTPL; 2) because of this, cost perturbations are not i.i.d. across experts, i.e. scheduling policies; 3) the Whittle index policy is only an approximate minimizer rather than an exact minimizer of the average AoI cost. Despite these challenges, we are able to show that FPWL achieves low regret compared to any fixed scheduling policy, if the Whittle policy is "close" to the actual optimal policy. Theorem 2 describes an upper bound on the regret of FPWL when compared to the best fixed scheduling policy in hindsight. Our regret bounds are in terms of a parameter α that measures the closeness between the Whittle policy and an optimal policy. For a detailed definition of α see Appendix B.

Theorem 2. *Follow the perturbed Whittle leader (FPWL) based scheduling described by Algorithm 2 with* $\epsilon = \sqrt{\frac{2M}{ND^2T}}$ *achieves the following upper bound on expected regret:*

$$\mathbb{E}[Regret_T(FPWL)] \leq \alpha T + 2D\sqrt{2MNT},$$

where the expectation is taken over the random perturbations.

Proof. See Appendix D in the technical report [36]. □

It was proved in [35] that the Whittle index policy is optimal for $N = 2$, meaning $\alpha = 0$ and we can achieve sublinear regret with respect to the best fixed scheduling policy when there are 2 sources. Further, recent work in [27] suggests that $\alpha \to 0$ as $N \to \infty$ meaning that FPWL can achieve sublinear regret for large system sizes. Simulations in both [35] and [27] indicate that α is very small for most problems of practical interest.

Importantly, note that there is no way to get sublinear static regret by using FPWL if the Whittle solution is not exactly optimal for the offline problem. In this case, even if the cost functions are the same in every epoch, there would be a small gap $\alpha > 0$ between the cost of the Whittle policy and the optimal policy in every epoch. The small constant gap will add up to give linear regret. Thus, the term αT in the regret bound above accounts for this cost of using an approximate optimization oracle rather than an exact one, and cannot be eliminated.

3.1.2 *Dynamic Regret.*

A drawback of the online learning formulation is that sublinear regret is only possible when comparing to a simple class of policies since there are no constraints on the adversary choosing the cost functions. A more general notion of regret is *dynamic regret* where cost is compared to an algorithm which chooses the best scheduling policy in *each* epoch rather than the best fixed policy across epochs. Dynamic regret of an algorithm that chooses scheduling policy π_k in epoch k is defined as follows:

$$\text{D-Regret}_T(\text{Alg.}, C) \triangleq \sup_{C_{1,..,T} \in C} \left\{ \sum_{k=1}^{T} C_k(\pi_k) - \sum_{k=1}^{T} \min_{\pi \in \Pi} C_k(\pi) \right\}, \tag{15}$$

where C incorporates constraints on the adversary. It is easy to show that if there are no constraints on how an adversary is allowed to choose the cost functions $C_1, ..., C_T$ then achieving sublinear dynamic regret is not possible. Thus, the definition of dynamic regret includes C which is the class of cost function sequences over which the regret is being considered and incorporates constraints on the adversary.

A number of recent works on online learning consider the problem of minimizing dynamic regret by constraining how the sequence of cost functions change over time (see [4, 5, 9, 16]). We follow the approach of [4] and [5] by defining the quantity V_T which measures the variation of a given sequence of cost functions as follows:

$$\sum_{k=2}^{T} \max_{\pi} \left| C_{k-1}(\pi) - C_k(\pi) \right| \leq V_T. \tag{16}$$

Suppose we know that any sequence of cost functions chosen by the adversary is going to satisfy the inequality (16). Then, we denote the set of allowable sequence of cost functions by $C(V_T)$ and define the quantity V_T as the variation budget given to the adversary.

We can also use the Whittle index approach to achieve low dynamic regret. If V_T is known to be sublinear in T beforehand, then simply using the Whittle index policy for the cost functions revealed in the previous epoch is sufficient to get low dynamic regret. Specifically, set $f_0^{(i)} = \{1, ..., M\}, \forall i \in \{1, ..., N\}$ and let the scheduling policy in epochs k be given by:

$$\pi_k = \text{Whittle}\left(f_{k-1}^{(1)}, ..., f_{k-1}^{(N)}\right). \tag{17}$$

We call this algorithm *Follow the Dynamic Whittle Leader* (FDWL).

Lemma 4. *The dynamic regret of FDWL satisfies*

$$D\text{-}Regret_T(FDWL, C(V_T)) \leq \alpha T + V_T + D,$$

where V_T *is the variation budget as defined in* (16) *and* D *is the upper-bound on AoI cost functions.*

Proof. See Appendix E in the technical report [36]. □

An important point to note here is that FDWL should only be used when an upper bound on V_T that grows sublinearly with T is known *a priori.* If no such upper bound is known and V_T grows linearly with T, then it can be shown that FDWL incurs static regret that is linear in T, meaning that it performs worse than FPWL (Algorithm 2). This neatly splits the full feedback setting into two regimes. If V_T is known to be sublinear use FDWL to get sublinear dynamic regret. Otherwise, use the entire history of cost functions as in FPWL to get sublinear static regret.

Algorithm 2 also highlights the strength of follow-the-leader style algorithms in solving online optimization problems with combinatorial structure. If a low complexity solution is known to the offline problem, as with the Whittle index, then it can be incorporated into FTPL as an optimization oracle.

3.1.3 *Bandit Feedback.*

For bandit feedback, the cost function of AoI associated with source i is only revealed during the time-slots in which it sends an update.Specifically, if at time-slot j within epoch k the policy π_k schedules sensor i, then $f_k^{(i)}(A^{(i)}(j))$ is revealed to the scheduler. This happens for every time-slot in the epoch.

Exponential weight update based algorithms like EXP3 [1] or EXP3.S [5] are standard in the bandit feedback case. Unlike follow-the-leader style algorithms, incorporating a Whittle index solution directly is not possible in these algorithms since they do not involve an explicit optimization step. This makes designing computationally efficient online learning algorithms for bandit feedback harder in the multiple source setting. We develop a heuristic solution for this below using our algorithms for the full feedback setting.

To run FPWL and FDWL on incomplete feedback we need to construct estimates of the cost functions denoted by $\hat{f}_k^{(1)}, ..., \hat{f}_k^{(N)}$. We do this by linearly interpolating between the revealed values of $f_k^{(i)}$ for each source i. Algorithm 3 describes the details. Importantly, constructing the linear interpolating cost estimates for a single source requires a single pass over $1, ..., M$. Thus, constructing $\hat{f}_k^{(1)}, ..., \hat{f}_k^{(N)}$ has a computational complexity $O(NM)$. So, our modified versions of FPWL and FDWL for bandit feedback remain computationally efficient. However, since these estimates are not guaranteed to be unbiased, regret analysis in the bandit feedback case becomes even more challenging. We leave this to future work.

Algorithm 3: Linearly Interpolating Cost Function Estimate for source i

```
Input  : $X \subseteq \{1, ..., M\}$ for which $f_k^{(i)}$ is known, $D$ known
         upper bound on $f_k^{(i)}$
Output : Estimate $\hat{f}_k^{(i)}$ that is an increasing AoI cost
         function
1  Add 0 to $X$ and set $f_k^{(i)}(0) = 0$
2  if $M \notin X$ then
3      set $f_k^{(i)}(M) = D$ and add $M$ to $X$
4  end
5  Sort $X$ in increasing order $\{0, x_1, ..., x_l, M\}$
6  while $h \in 1, ..., M$ do
7      if $h \notin X$ then
8          Find $k$ such that $x_k < h < x_{k+1}$ and $x_k, x_{k+1} \in X$
9          Set $\hat{f}_k^{(i)}(h) = f_k^{(i)}(x_k) + (h - x_k)\frac{f_k^{(i)}(x_{k+1}) - f_k^{(i)}(x_k)}{x_{k+1} - x_k}$
10     else
11         Set $\hat{f}_k^{(i)}(h) = f_k^{(i)}(h)$.
12     end
13 end
```

4 MOBILITY TRACKING

We now apply the results we have developed to a mobility tracking problem. Consider N nodes moving around in the two dimensional plane whose positions needs to be tracked by a central base station (BS). At any given time, only one of these nodes can send an update about its current location and velocity to the BS. The BS keeps track of the location of the nodes by storing the most recently received update from each node. Our goal is to design a scheduling policy that minimizes total tracking error between the location estimates at the BS and the actual locations of the nodes.

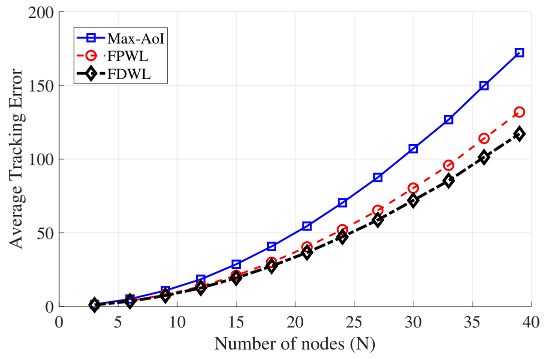

Figure 3: Levy Mobility: Average Tracking Error v/s number of nodes

Observe that if the current velocity of a node i is v_i, then its tracking error grows linearly with its AoI. That is, if the BS hasn't received an update from node i for time A_i, then the tracking error is $v_i A_i$. In practical scenarios, node mobility patterns and velocities are often unknown beforehand, non-stationary, and possibly adversarial. Thus, our mobility tracking problem can be viewed as a weighted-AoI minimization problem with time-varying velocities acting as weights. Since a new update from a node only contains information about its current location and velocity, *this fits into the multiple source bandit feedback setting.*

We will discuss two specific mobility models and apply our online algorithms to show that they outperform static AoI based scheduling. Note that while cost functions being static within an epoch and resetting of AoIs at the beginning of every epoch are necessary for regret analysis, these assumptions are not required to implement our algorithms in practice.

4.1 Levy Mobility

In this scenario, we simulate the nodes' motion using Levy mobility. This is a realistic mobility model that closely matches human mobility in practice [29]. A node's motion is described in a series of *steps*. A step is represented by the tuple (v, θ, t_f, t_p) - a velocity v picked randomly in the interval $[0, v_{\max}]$, an angle θ picked uniformly from the interval $[0, 2\pi]$, a flight time t_f picked at random from $\{1, ..., T_{f_{\max}}\}$ and a pause time t_p picked at random from $\{1, ..., T_{p_{\max}}\}$. The node then moves with the velocity v, in the direction θ for t_f time-slots and then pauses at its location for t_p time-slots. This leads to a bursty random walk pattern with time-varying velocities.

We consider N nodes executing Levy mobility. An adversary sets the values of $v_{\max}$ for each node from the set $\{0.1, 0.5, 5\}$ designating it as a slow, medium or fast node. Overall, $N/3$ nodes each are designated as fast, medium and slow, but the scheduler doesn't know which. We set $T_{f_{\max}} = 50$ and $T_{p_{\max}} = 30$ for all nodes.

The scheduler does not know beforehand that there is inherent asymmetry in the motion of the nodes. An oblivious static scheduling policy is max-AoI: let the node with the maximum AoI transmit in every time-slot. From Figure 3, we observe that using FPWL in this setting outperforms the max-AoI scheduling policy (by about 25%). Further, FDWL outperforms both max-AoI (by about 33%)

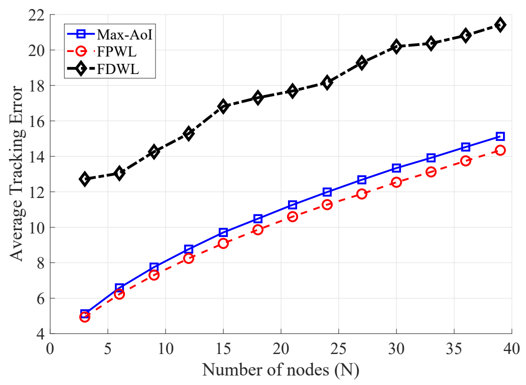

Figure 4: Adversarial Mobility: Average Tracking Error v/s number of nodes

and FPWL (by about 10%). This is because velocities under Levy mobility are slowly varying in time and not adversarial, allowing a dynamic regret based algorithm such as FDWL to work better than FPWL. We set the epoch length M to 200 time-slots for both FPWL and FDWL, and the number of epochs T to 500, thus running the simulation for 100000 time-slots.

4.2 Adversarial Mobility

In this scenario, we assume that the nodes execute a mobility pattern that is chosen by a reactive adversary in response to the scheduling policies. In every epoch, the nodes execute Brownian motion (moving in random directions at a fixed velocity). An adversary assigns velocities to nodes such that they are inversely proportional to their scheduling priorities.

For FPWL, the scheduling policy in epoch t is given by π_t = Whittle$\left(F_t^{(1)} + \gamma_t^{(1)}, ..., F_t^{(N)} + \gamma_t^{(N)}\right)$. So, the velocity $v_t^{(i)}$ of node i in epoch t is chosen to satisfy $v_k^{(i)} \propto c^{(i)}||F_t^{(i)}||^{-2}$. Similarly, for FDWL, the scheduling policy in epoch t is given by π_t = Whittle$\left(\hat{f}_{t-1}^{(1)}, ..., \hat{f}_{t-1}^{(N)}\right)$, where we use estimated cost functions since our setting involves bandit feedback. So, $v_t^{(i)}$ is chosen to satisfy $v_k^{(i)} \propto c^{(i)}||\hat{f}_{t-1}^{(i)}||^{-2}$. For the max-AoI policy, the velocity $v_t^{(i)}$ is chosen to satisfy $v_k^{(i)} \propto c^{(i)}$. Here $c^{(i)}$ are parameters which are fixed across epochs and also chosen by the adversary to ensure that the motion of nodes has inherent asymmetry unknown to the scheduler. Overall, $N/3$ nodes each are assigned $c^{(i)} = 0.1$, $c^{(i)} = 0.4$ and $c^{(i)} = 40$. If the scheduler observes a node was moving fast in the previous epochs and assigns it a larger cost, then the adversary assigns it a low velocity in the next epoch so as to confuse the scheduler. The sum total of velocities is normalized and remains fixed in every time-slot ensuring that the *adversary is equally powerful irrespective of scheduling policies.*

Under this adversarial model, we observe in Figure 4 that while FPWL still outperforms max-AoI (by about 8%), FDWL performs significantly worse than both FPWL and max-AoI (about 50% worse). This is consistent with our results from theory - when cost functions are quickly varying and adversaries are unconstrained and reactive, dynamic regret based algorithms like FDWL perform worse than static regret algorithms like FPWL.

5 CONCLUSION

In this work, we have formulated a general framework for online monitoring and scheduling for non-stationary sources. Specifically, we handle unknown, time-varying, and possibly adversarial cost functions of AoI and design algorithms that attempt to learn the best scheduling policies in an online fashion. We apply our results to a mobility tracking problem and show that our online learning algorithms outperform oblivious AoI based schemes and are able to learn information about the underlying source dynamics.

Possible directions of future work involve applying our online scheduling framework to different problems of practical interest, and incorporating unreliable channels and noisy feedback about the costs into our framework.

ACKNOWLEDGMENTS

This work was supported by NSF Grant CNS-1713725, and by Army Research Office (ARO) grant number W911NF-17-1-0508.

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.* 32, 1 (2002), 48–77.
[2] Subhankar Banerjee, Rajarshi Bhattacharjee, and Abhishek Sinha. 2020. Fundamental limits of age-of-information in stationary and non-stationary environments. *arXiv preprint arXiv:2001.05471* (2020).
[3] Ahmed M Bedewy, Yin Sun, and Ness B Shroff. 2019. Minimizing the age of information through queues. *IEEE Trans. Information Theory* 65, 8 (2019), 5215–5232.
[4] Omar Besbes, Yonatan Gur, and Assaf Zeevi. 2015. Non-stationary stochastic optimization. *Operations research* 63, 5 (2015), 1227–1244.
[5] Omar Besbes, Yonatan Gur, and Assaf Zeevi. 2019. Optimal exploration–exploitation in a multi-armed bandit problem with non-stationary rewards. *Stochastic Systems* 9, 4 (2019), 319–337.
[6] Kavya Bhandari, Santosh Fatale, Urvidh Narula, Sharayu Moharir, and Manjesh Kumar Hanawal. 2020. Age-of-Information Bandits. *arXiv preprint arXiv:2001.09317* (2020).
[7] Nicolo Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games.* Cambridge university press.
[8] Jaya Prakash Champati, Mohammad H Mamduhi, Karl H Johansson, and James Gross. 2019. Performance characterization using aoi in a single-loop networked control system. In *Proc. IEEE INFOCOM AoI Workshop.* 197–203.
[9] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. 2019. Learning to optimize under non-stationarity. In *Proc. Int. Conf. Artificial Intell. Stats. (AISTATS).* 1079–1087.
[10] Alon Cohen and Tamir Hazan. 2015. Following the perturbed leader for online structured learning. In *Proc. Int. Conf. Machine Learning (ICML).* 1034–1042.
[11] Shahab Farazi, Andrew G Klein, John A McNeill, and D Richard Brown. 2018. On the age of information in multi-source multi-hop wireless status update networks. In *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC).* 1–5.
[12] James Hannan. 1957. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games* 3 (1957), 97–139.
[13] Elad Hazan. 2019. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207* (2019).
[14] Longbo Huang and Eytan Modiano. 2015. Optimizing age-of-information in a multi-class queueing system. In *Proc. IEEE Int. Symp. Information Theory (ISIT).* 1681–1685.
[15] Yoshiaki Inoue, Hiroyuki Masuyama, Tetsuya Takine, and Toshiyuki Tanaka. 2018. A general formula for the stationary distribution of the age of information and its application to single-server queues. *arXiv preprint arXiv:1804.06139* (2018).
[16] Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. 2015. Online optimization: Competing with dynamic comparators. In *Proc. Int. Conf. Artificial Intell. Stats. (AISTATS).* 398–406.
[17] Prakirt Raj Jhunjhunwala and Sharayu Moharir. 2018. Age-of-Information Aware Scheduling. In *Proc. IEEE SPCOM.*
[18] Igor Kadota, Abhishek Sinha, and Eytan Modiano. 2019. Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints. *IEEE/ACM Trans. Netw.* 27, 4 (2019), 1359–1372.

[19] Igor Kadota, Abhishek Sinha, Elif Uysal-Biyikoglu, Rahul Singh, and Eytan Modiano. 2018. Scheduling policies for minimizing age of information in broadcast wireless networks. *IEEE/ACM Trans. Netw.* 26, 6 (2018), 2637–2650.

[20] Adam Kalai and Santosh Vempala. 2005. Efficient algorithms for online decision problems. *J. Comput. System Sci.* 71, 3 (2005), 291–307.

[21] Clement Kam, Sastry Kompella, and Anthony Ephremides. 2013. Age of information under random updates. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*. 66–70.

[22] Clement Kam, Sastry Kompella, and Anthony Ephremides. 2019. Learning to sample a signal through an unknown system for minimum aoi. In *Proc. IEEE INFOCOM AoI Workshop*. 177–182.

[23] Sanjit Kaul, Roy Yates, and Marco Gruteser. 2012. Real-time status: How often should one update?. In *Proc. IEEE INFOCOM*. 2731–2735.

[24] Markus Klügel, Mohammad H Mamduhi, Sandra Hirche, and Wolfgang Kellerer. 2019. Aoi-penalty minimization for networked control systems with packet loss. In *Proc. IEEE INFOCOM AoI Workshop*. 189–196.

[25] Antzela Kosta, Nikolaos Pappas, Vangelis Angelakis, et al. 2017. Age of information: A new concept, metric, and tool. *Foundations and Trends in Networking* 12, 3 (2017), 162–259.

[26] Antzela Kosta, Nikolaos Pappas, Anthony Ephremides, and Vangelis Angelakis. 2017. Age and value of information: Non-linear age case. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*. 326–330.

[27] Ali Maatouk, Saad Kriouile, Mohamad Assaad, and Anthony Ephremides. 2020. On The Optimality of The Whittle's Index Policy For Minimizing The Age of Information. *arXiv preprint arXiv:2001.03096* (2020).

[28] Tasmeen Zaman Ornee and Yin Sun. 2019. Sampling for remote estimation through queues: Age of information and beyond. *IEEE Int. Symp. Model. Optim. Mobile, Ad Hoc Wireless Netw. (WiOpt)* (2019).

[29] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seong Joon Kim, and Song Chong. 2011. On the levy-walk nature of human mobility. *IEEE/ACM Trans. Netw.* 19, 3 (2011), 630–643.

[30] Yin Sun and Benjamin Cyr. 2019. Sampling for data freshness optimization: Non-linear age functions. *IEEE Journal Commun. Netw.* 21, 3 (2019), 204–219.

[31] Yin Sun, Igor Kadota, Rajat Talak, and Eytan Modiano. 2019. Age of information: A new metric for information freshness. *Synthesis Lectures on Communication Networks* 12, 2 (2019), 1–224.

[32] Yin Sun, Yury Polyanskiy, and Elif Uysal-Biyikoglu. 2017. Remote estimation of the Wiener process over a channel with random delay. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*. 321–325.

[33] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff. 2017. Update or Wait: How to Keep Your Data Fresh. *IEEE Trans. Information Theory* 63, 11 (Nov. 2017), 7492–7508.

[34] Rajat Talak, Sertac Karaman, and Eytan Modiano. 2018. Optimizing information freshness in wireless networks under general interference constraints. In *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*. 61–70.

[35] Vishrant Tripathi and Eytan Modiano. 2019. A whittle index approach to minimizing functions of age of information. In *Proc. 57th Allerton Conf. Commun. Control Comput.* IEEE, 1160–1167.

[36] Vishrant Tripathi and Eytan Modiano. 2020. *An Online Learning Approach to Optimizing Time-Varying Costs of AoI.* Technical Report. https://arxiv.org/abs/2105.13383.

[37] Vishrant Tripathi and Sharayu Moharir. 2017. Age of information in multi-source systems. In *Proc. IEEE Global Commun. Conf. (GLOBECOM)*. 1–6.

[38] Jihyeon Yun, Changhee Joo, and Atilla Eryilmaz. 2018. Optimal real-time monitoring of an information source under communication costs. In *IEEE Conf. Decis. Control (CDC)*. 4767–4772.

[39] Xi Zheng, Sheng Zhou, and Zhisheng Niu. 2019. Context-aware information lapse for timely status updates in remote control systems. In *Proc. IEEE Global Commun. Conf. (GLOBECOM)*. 1–6.

A PROOF OF LEMMA 2

Let the AoI cost function in epoch k be $f_k(\cdot)$, let the transmission cost be C and let the chosen sampling threshold be x. We set $t = 1$ at the beginning of the epoch. Then,

$$C_k(x) = \sum_{t=1}^{M} f_k(A(t)) + Cu(t). \tag{18}$$

Note that the AoI at time $t = 1$ is $A(1) = 1$, since each epoch begins after a new transmission. Since the threshold is set to x, no new update is sent till time-slot x at which point the AoI reaches x. Now, a new sample is generated and sent, so the AoI drops to 1 in the next time-slot. This process repeats in cycles of x time-slots. Since the epoch consists of M time-slots, there are $\lfloor \frac{M}{x} \rfloor$ complete cycles of length x. The sum of costs over each of these cycles is $\big(\sum_{j=1}^{x} f_k(j) + C\big)$ since the AoI goes from 1 to x and there is a transmission at the end.

The final cycle is of length $r = M \mod x$, where $a \mod b$ is the remainder when a is divided by b, followed by a mandatory transmission in the final time-slot. Thus,

$$C_k(x) = \left\lfloor \frac{M}{x} \right\rfloor \left(\sum_{j=1}^{x} f_k(j) + C \right) + \mathbb{1}_{r>0} \left(\sum_{j=1}^{r} f_k(j) + C \right). \tag{19}$$

B CLOSENESS OF WHITTLE AND OPTIMAL POLICIES

Here, we define α, the parameter that measures the closeness of the Whittle index policy to an optimal policy within an epoch.

Consider a set of monotone and bounded AoI cost functions $f^{(1)}, ..., f^{(N)}$ such that for all i, if $x > y$ then $f^{(i)}(x) \geq f^{(i)}(y)$ and $f^{(i)}(M) \leq D$. Let Whittle(f) denote the Whittle policy for this set of cost functions, as defined in (14). Let Opt(f) denote an optimal policy for this set of cost functions.

Now consider another set of monotone bounded AoI cost functions $g^{(1)}, ..., g^{(N)}$ with the same upper bound D. Given a scheduling policy π, let

$$C_g(\pi) \triangleq \frac{1}{NM} \sum_{j=1}^{M} \sum_{i=1}^{N} g^{(i)}(A^{(i)}(j)), \tag{20}$$

where the AoIs evolve under policy π. This is the total sum cost of policy π under the cost functions $g^{(1)}, ..., g^{(N)}$. We make the following assumption on the structure of Whittle index and optimal policies when the epoch length M is long.

Assumption 1. *For any two sets of bounded monotone sets of cost functions $f^{(1)}, ..., f^{(N)}$ and $g^{(1)}, ..., g^{(N)}$ with a fixed known upper bound D, the following holds:*

$$\Big| C_g\big(Whittle(f)\big) - C_g\big(Opt(f)\big) \Big| \leq \alpha, \tag{21}$$

where α is a small constant that can depend on N, M and D.

Note that this assumption is stronger than just assuming that the Whittle index policy has near optimal performance over long epochs. We assume that the Whittle policy is also close to the optimal policy in its sequence of scheduling decisions. Thus, given arbitrary bounded cost functions, the two policies $C_g\big(\text{Whittle}(f)\big)$ and $C_g\big(\text{Opt}(f)\big)$ have average costs that are close to each other. This is a Lipschitz like assumption on the policy space and cost functions for the scheduling problem. The motivation for this comes from results in [35], where it was shown that the Whittle policy is exactly optimal for $N = 2$ as $M \to \infty$, meaning that we can set $\alpha = 0$. It was also observed via simulations that the Whittle policies are structurally similar to optimal policies for general N. Results on asymptotic optimality of the Whittle policy [27] further suggest that $\alpha \to 0$ as $N \to \infty$.