ABSTRACT

Title of dissertation: AGE OF INFORMATION OPTIMIZATION FOR TIMELINESS IN COMMUNICATION NETWORKS

Melih Bastopcu, Doctor of Philosophy, 2021

Dissertation directed by: Professor Sennur Ulukus
Department of Electrical and Computer Engineering

With the emergence of technologies such as autonomous vehicular systems, holographic communications, remote surgery and high frequency automated trading, timeliness of information has become more important than ever. Most traditional performance metrics, such as delay or throughput, are not sufficient to measure timeliness. For that, age of information (AoI) has been introduced recently as a new performance metric to quantify the timeliness in communication networks. In this dissertation, we consider timely update delivery problems in communication networks under various system settings.

First, we introduce the concept of soft updates, where different from the existing literature, here, updates are soft and begin reducing the age immediately but drop it gradually over time. Our setting models human interactions where updates are soft, and also social media interactions where an update consists of viewing and digesting many small pieces of information posted, that are of varying importance, relevance and interest to the receiver. For given total system duration, the number

of updates, and the total allowed update duration, we find the optimum start times of the soft updates and their optimum durations to minimize the overall age.

Then, we consider an information updating system where not only the timeliness but also the quality of the updates is important. Here, we use distortion as a proxy for quality, and model distortion as a decreasing function of processing time spent while generating the updates. Processing longer at the transmitter results in a better quality (lower distortion) update, but it causes the update to age in the process. We determine age-optimal policies by characterizing the update request times at the receiver and the update processing times at the transmitter subject to constant or age-dependent distortion constraints on each update.

Next, different from most of the existing literature on AoI where the transmission times are based on a given distribution, by assigning codeword lengths for each status update, we design transmission times through source coding schemes. In order to further improve timeliness, we propose selective encoding schemes where only the most probable updates are transmitted. For the remaining least probable updates, we consider schemes where these updates are never sent, randomly sent, or sent by an empty symbol. For all these encoding schemes, we determine the optimal number of encoded updates and their corresponding age-optimal real-valued codeword lengths to minimize the average age at the receiver.

Then, we study the concept of generating partial updates which carry less information compared to the original updates, but their transmission times are shorter. Our aim is to find the age-optimal partial update generation process and the corresponding age-optimal real-valued codeword lengths for the partial updates while maintaining a desired level of fidelity between the original and partial updates.

Next, we consider information freshness in a cache updating system consisting of a source, cache(s) and a user. Here, the user may receive an outdated file depending on the freshness status of the file at the cache. We characterize the binary freshness metric at the end user and propose an alternating maximization based method to optimize the overall freshness at the end user subject to total update rate constraints at the cache(s) and the user.

Then, we study a caching system with a limited storage capacity for the cache. Here, the user either gets the files from the cache, but the received files can be sometimes outdated, or gets fresh files directly from the source at the expense of additional transmission times which inherently decrease the freshness. We show that when the total update rate and the storage capacity at the cache are limited, it is optimal to get the frequently changing files and files with relatively small transmission times directly from the source, and store the remaining files at the cache.

Next, we focus on information freshness in structured gossip networks where in addition to the updates obtained from the source, the end nodes share their local versions of the updates via gossiping to further improve freshness. By using a stochastic hybrid systems (SHS) approach, we determine the information freshness in arbitrarily connected gossip networks. When the number of nodes gets large, we find the scaling of information freshness in disconnected, ring and fully connected network topologies. Further, we consider clustered gossip networks where multiple clusters of structured gossip networks are connected to the source through cluster heads, and find the optimal cluster sizes numerically.

Then, we consider the problem of timely tracking of multiple counting random processes via exponential (Poisson) inter-sampling times, subject to a total sampling rate constraint. A specific example is how a citation index such as Google Scholar should update citation counts of individual researchers to keep the entire citation index as up-to-date as possible. We model citation arrival profile of each researcher as a counting process with a different mean, and consider the long-term average difference between the actual citation numbers and the citation numbers according to the latest updates as a measure of timeliness. We show that, in order to minimize this difference metric, Google Scholar should allocate its total update capacity to researchers proportional to the square roots of their mean citation arrival rates.

Next, we consider the problem of timely tracking of multiple binary random processes via sampling rate limited Poisson sampling. As a specific example, we consider the problem of timely tracking of infection status (e.g., covid-19) of individuals in a population. Here, a health care provider wants to detect infected and recovered people as quickly as possible. We measure the timeliness of the tracking process as the long term average difference between the actual infection status of people and their real-time estimate at the health care provider which is based on the most recent test results. For given infection and recovery rates of individuals, we find the exponentially applied testing rates for individuals to minimize this difference. We observe that when the total test rate is limited, instead of applying tests to everyone, only a portion of the population should be tested.

Finally, we consider a communication system with multiple information sources that generate binary status updates, which in practical application may indicate an anomaly (e.g., fire) or infection status (e.g., covid-19). Each node exhibits an anomaly or infection with probability p. In order to send the updates generated by these sources as timely as possible, we propose a group updating method inspired by group testing, but with the goal of minimizing the overall average age, as opposed to the average number of tests (updates). We show that when the probability p is small, group updating method achieves lower average age than the sequential updating methods.

Age of Information Optimization for Timeliness in Communication Networks

by

Melih Bastopcu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Dedication

To my mother Sakire Bastopcu for her endless support.

Acknowledgments

I gratefully thank my advisor Professor Sennur Ulukus for her all help and support during my Ph.D. studies. With her invaluable feedback on my papers, presentations, and teaching styles, I can confidently say that I owe everything related to my professional and academic development to her. I am very impressed with her personal involvement in our research problems and her endless passion on new research directions. She is always open to new ideas and provides freedom for us to come up with new research directions. Before covid-19, her door used to be always open for our research meetings which I enjoyed a lot. Even with covid-19, she was always accessible via Zoom for late hours. My personal record was our Zoom meeting that ended around 11:20 pm. Even though Ph.D. program can be sometimes stressful with the challenging classes, qualification exams, obtaining results in our research problems, I inherently knew that she was going to support me which made me feel always comfortable. I am also thankful for her genuine supports during my academic position applications.

I would like to thank Professors Adrian Papamarcou, Prakash Narayan, Nuno Martins, Radu Balan, for being in my dissertation committee and offering their valuable feedback. I would like to also thank Professor Antony Ephremides for his valuable comments on my proposal. I am thankful to all the professors I have interacted with over the past years at UMD. I especially want to thank Professors Prakash Narayan and Nuno Martins whom I enjoyed their classes and I was honored to be their teaching assistant over the years.

I am thankful to all my lab mates at UMD. I would like to specifically thank Baturalp Buyukates, Batuhan Arasli, Brian Kim, Priyanka Kaswan, Matin Mortaheb, Cemil Vahapoglu, Sajani Vithana, Zhusheng Wang, Purbesh Mitra, Mustafa Doger, Dr. Yi-Peng Wei, Dr. Karim Banawan, Dr. Abdulrahman Baknina, Dr. Ahmed Arafa, Dr. Pritam Mukherjee, Dr. Berk Gurakan, Ajaykrishnan Nageswaran, Sagnik Bhattacharya, Dr. Vinay Praneeth Boda.

I personally thank my dearest friend, lab-mate, and (then became) a roommate, Baturalp Buyukates, who was always with me for the hardest and as well as the happiest moments of this long journey. I feel very lucky that we started this program together and we bear all the challenges of Ph.D. studies together. I always relied on his honest views in every part of my academic and personal life. He was very thoughtful to me and I will always miss his knocking on my door during a day. I am also very thankful to my other roommates, Batuhan Arasli and Ozde Ozkaya, for making our house feel like home. They were like an extended family to me. I am specially thankful for my friends at UMD, Semih Kara, Ece Yegane, Hulya Biler, Gamze Yavuzer, Siddharth Tyagi, Deepayan Bhadra, and my friends from college, Burak Bartan, and Umitcan Sahin, and my friends from high school, Harun Avci, Nurullah Ishak Isik, and Mustafa Bugrahan Gurbuz.

My deepest gratitude goes to my family for all their support throughout my life. I would like to specially thank my mother Sakire Bastopcu who always supported me in any possible direction, my sister Gozde Bezirgan, who always tried to convince me to go back to Turkey, my brother-in-law Selman Bezirgan, my little lovely nephew Ekin Yagmur, my grandmother Muruvvet Ezerel, who has recently passed away, my aunt Meral Ezerel Temel, and her husband Ishak Temel, my lovely cousins Yigit and Elif Naz.

Finally, I would like to thank the ECE staff members, who are always dedicated to help all graduate students with their sincere efforts. In particular, I would like to specially thank to Melanie Prange who is very helpful and knowledgeable, and always has answers to any of my concerns throughout my studies, Emily Irwin, Maria Hoo, and Vivian Lu.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Timely information delivery in communication networks has been recognized to be important, especially in light of emerging applications, such as autonomous vehicular systems, holographic communications, remote surgery, and high frequency automated trading. Common to all these applications is the fact that information is most useful when it is freshest. Traditional network design metrics, such as delay and throughput, do not capture information timeliness. Information timeliness requires a careful combination of sufficiently low delay and sufficiently high throughput. For example, even though the updates can be delivered with low delay, if the inter update generation times are large (i.e., if the throughput is low), critical changes in the status of the system may be missed between two status updates. Similarly, even though a large throughput can be achieved by sending a large number of status updates, due to possible waiting times and transmission delays as a result of congestion, the received updates may already be outdated. Thus, we need a different performance metric that is capable of measuring information timeliness in communication systems.

Although various concepts of timeliness have appeared in the literature before, such as in the context of web freshness in [1], recent literature on timeliness in communication networks has started with the *age of information (AoI)* metric defined in [2]. This, now conventional, timeliness metric is defined as the time elapsed since the generation of the most recently received status update at the receiver. As shown in Fig. 1.1(a), with this metric, age at the receiver increases linearly over time when no updates are delivered; and decreases down to the age of the most recently received status update upon a status update delivery. Age of information has been studied in queueing systems [2–14], multi-hop and multi-cast networks [15–25], social networks [26], content freshness in the web [1,27–29], timely remote estimation of random processes [30–36], energy harvesting systems [37–55], wireless fading channels [56,57], scheduling in networks [58–76], lossless and lossy source and channel coding [77–87], vehicular, IoT and UAV systems [88–91], caching systems [92–103], computation-intensive systems [104–113], learning and Markov decision process settings [114–128], and so on. A more detailed review of the age of information literature can be found in references [129,130].

The conventional age metric defined in [2] is useful in characterizing information timeliness in many systems, but may have limitations in some other applications. With the conventional age metric, information at the receiver becomes stale when there are no update deliveries (as the age increases linearly over time). However, if the information at the source has not changed, even though the receiver did not receive any updates for some time, the information at the receiver may still be the freshest (e.g., see in Fig. 1.1(b) that the receiver has the highest freshness

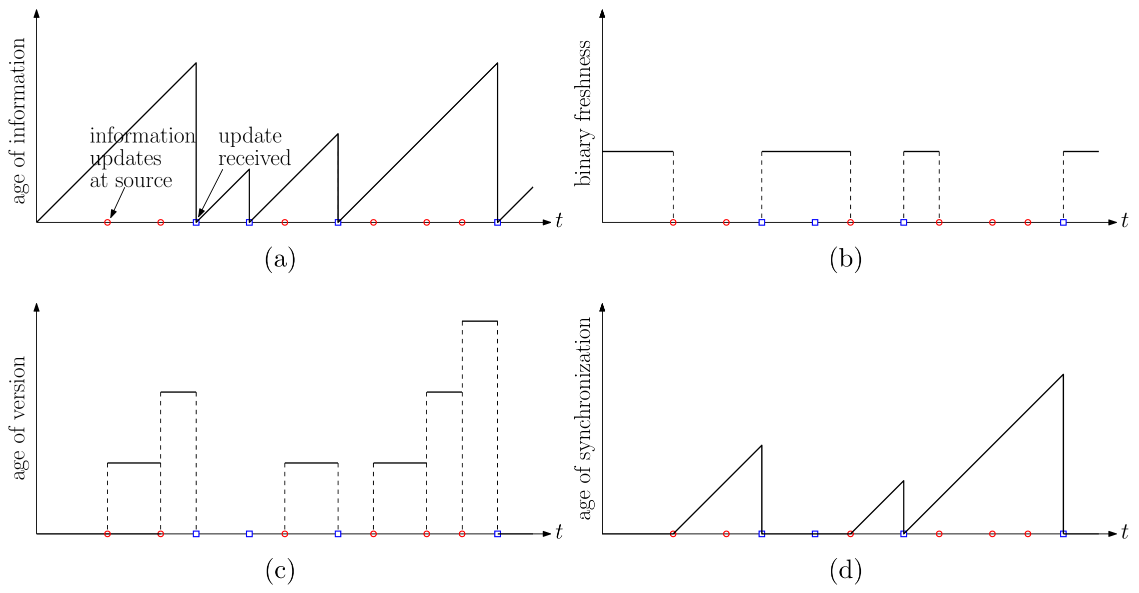


Figure 1.1: Sample evolution of (a) the traditional AoI, (b) the binary freshness, (c) the age of version, and (d) the age of synchronization (special case of AoII) metrics are provided. Here, red circles represent the update arrivals at the source, blue squares represent the updates received by the receiver. For simplicity, we assume that the transmission times are equal to 0.

until the first information update at the source, represented by a red circle, arrives as opposed to Fig. 1.1(a) where age continues to increase until a blue square arrives). Since the conventional age metric does not incorporate information variation rates at the source, it does not capture this modified sense of information freshness. For that, recently, several variations of the conventional age metric have been introduced. One of such metrics is the *binary freshness metric (BFM)*, which takes the value 1 when the information is fresh, i.e., the receiver has the same version as the source, and the value 0, when the information at the receiver is different from the source [1, 27, 28, 99, 101, 102]. Another such metric is the *age of version (AoV)* metric. If we view each information change at the source as a new version, age of version, shown in Fig. 1.1(c), increases by 1 whenever the version changes at the source and decreases down to 0 when the receiver gets the current version at the source. In other words, AoV measures how many versions an information receiver is behind compared to the currently prevailing version at the source [131–133]. A more general metric that considers information variations at the source and also has a time component is the *age of incorrect information (AoII)* which stays as 0 as long as information at the receiver is fresh and may increase over time when the information at the receiver is stale [34, 70, 134]. With that, *age of synchronization (AoS)* introduced in [97] and shown in Fig. 1.1(d) can be considered as a special case of AoII.

In this dissertation, different from most of the works in the AoI literature which consider the conventional age as the only performance metric, we consider timeliness metrics that incorporate information change statistics at the source, along with the concepts of soft updates, quality of updates, and partial updates, to develop fundamental solutions for timely update delivery problems in communication networks, by utilizing tools from optimization, wireless communication, and information theory. In the remainder of this introduction, we describe each chapter of the dissertation in more detail.

In Chapter 2, different from the existing literature where updates are countable (hard) and take effect either immediately or after a delay, but instantaneously in both cases, we consider an information updating system where updates start taking effect right away but gradually over time. We coin this setting *soft updates*. When the updating process starts, the age decreases until the soft update period ends. We consider two models for the decrease of age during an update period: In the first model, the rate of decrease of age is proportional to the current age, and in the second model, the rate of decrease of age is constant. The first model results in an exponentially decaying age, and the second model results in a linearly decaying age. In both cases, we determine the optimum updating schemes, by determining the optimum start times and optimum durations of the updates, subject to the constraints on the number of update periods and the total update duration. For both models, in the optimal policy, we show that total update duration should be utilized by allocating equal amount of time for each update.

In Chapter 3, we study the timeliness of an information updating system along with the *distortion* on each update. Here, the updates are generated at the information provider (transmitter) as a result of completing a set of tasks such as collecting data and performing computations on them. We refer to this as the update generation process. We model distortion on the updates as a decreasing function of processing time spent while generating the updates at the transmitter. Processing longer at the transmitter results in a better quality (lower distortion) update, but it causes the update to age in the process. We determine the age-optimal policies for the update request times at the receiver and the update processing times at the transmitter subject to a minimum required quality (maximum allowed distortion) constraint on the updates. For the required quality constraint, we consider the cases of constant maximum allowed distortion constraints, as well as age-dependent maximum allowed distortion constraints. For all these distortion constraints, we show that the processing times are equal to the minimum required processing duration that meets the distortion constraint.

In Chapter 4, we consider a system where the information source generates independent and identically distributed status update messages from an observed random phenomenon which takes n distinct values based on a given probability mass function (pmf). These update packets are encoded at the transmitter node to be sent to a receiver node which wants to track the observed random variable as timely as possible. In order to further improve the timeliness of the system, the transmitter node implements a *selective k encoding policy* such that rather than encoding all possible n realizations, the transmitter node encodes the most probable k realizations. We consider three different policies regarding the remaining $n-k$ less probable realizations: *highest k selective encoding* which disregards whenever a realization from the remaining $n-k$ values occurs; *randomized selective encoding* which encodes and sends the remaining $n-k$ realizations with a certain probability to further inform the receiver node at the expense of longer codewords for the selected k realizations; and *highest k selective encoding with an empty symbol* which sends a designated empty symbol when one of the remaining $n-k$ realizations occurs. For all of these three encoding schemes, we find the average age and determine the age-optimal real codeword lengths, including the codeword length for the empty symbol in the case of the latter scheme, such that the average age at the receiver node is minimized. Through numerical evaluations for arbitrary pmfs, we show that these selective encoding policies result in a lower average age than encoding every realization, and find the corresponding age-optimal k values.

In Chapter 5, we study an information updating system where the transmitter further processes the updates obtained from the source in order to generate *partial* *updates*, which have smaller information compared to the original updates, to be sent to a receiver. We study the problem of generating partial updates, and finding their corresponding real-valued codeword lengths, in order to minimize the average age experienced by the receiver, while maintaining a desired level of mutual information between the original and partial updates. Since the original problem is NP hard, we relax the problem and develop an alternating minimization based iterative algorithm that generates a pmf for the partial updates, and the corresponding age-optimal real-valued codeword length for each update. We observe that there is a trade-off between the attained average age and the mutual information between the original and partial updates.

In Chapter 6, we consider the *binary freshness metric* in a *cache updating system* with a source, a cache and a user. The source keeps the freshest version of n files which are updated with known rates. The cache downloads and keeps the freshest version of the files from the source. The user gets updates from the cache. When the user gets an update, it either gets a fresh update from the cache or the file at the cache becomes outdated by a file update at the source in which case the user gets an outdated update. We find an analytical expression for the average binary freshness of the files at the user in terms of the file update rates at the source, at the cache, and at the user. Next, we generalize our setting to the case where there are multiple caches in between the source and the user, and find the average freshness at the user. We provide an alternating maximization based method to find the update rates for the cache(s), and for the user to maximize the freshness of the files at the user. We observe that for a given set of update rates for the user (resp. for the cache), the optimal rate allocation policy for the cache (resp. for the user) is a *threshold policy*, where the optimal update rates for rapidly changing files at the source may be equal to zero. In addition, in Chapter 6, we consider the case where multiple users are connected to a single cache and find update rates for the cache and the users to maximize the total freshness over all users.

In Chapter 7, different from the previous chapter, we study a cache updating system with a source, a *cache with limited storage capacity* and a user. Here, the cache gets fresh files from the source, but it can only store the latest downloaded versions of a subset of the files. The user gets the files either from the cache or from the source. If the user gets the files from the cache, the received files might be outdated depending on the file status at the source. If the user gets the files directly from the source, then the received files are always fresh, but the extra transmission times between the source and the user decreases the freshness at the user. Thus, in this chapter, we study the trade-off between storing the files at the cache and directly obtaining the files from the source at the expense of additional transmission times. We find analytical expressions for the average freshness of the files at the user for both of these scenarios. Then, we find the optimal caching status for each file (i.e., whether to store the file at the cache or not) and the corresponding file update rates at the cache to maximize the overall freshness at the user. We observe that when the total update rate of the cache is high, caching files improves the freshness at the user. However, when the total update rate of the cache is low, the optimal policy for the user is to obtain the frequently changing files and the files that have relatively small transmission times directly from the source.

In Chapter 8, we consider the binary freshness metric for *gossip networks* that consist of a single source and end-nodes, where the nodes are allowed to share their stored versions of the source information with the other nodes. First, we develop recursive equations that characterize the binary freshness in arbitrarily connected gossip networks by using a stochastic hybrid systems (SHS) approach. Next, we study the binary freshness in structured gossip networks and show that when the number of nodes becomes large, the binary freshness of a node decreases down to 0 as n^{-1} for the disconnected network topology (where the nodes are only connected to the source) and for the ring network topology (where the nodes are connected to two neighbor nodes) but with a strictly higher freshness for the ring network. The rate of decrease to 0 is slower for the fully connected networks (where each node is connected to every other node) when the update rates of the source as well as the end-nodes are sufficiently high. In addition, we study the binary freshness for clustered gossip networks, where multiple clusters of structured gossip networks are connected to the source through designated access nodes, i.e., cluster heads. We characterize the binary freshness in such networks and numerically observe how the optimal cluster sizes change with respect to the update rates of the source, cluster heads, and end-nodes.

In Chapter 9, we study the problem of real-time timely tracking of multiple *counting* processes via a resource-constrained Poisson updater, i.e., inter-sampling times are exponentially distributed. As a particular case, we consider Google Scholar, which wishes to update the citation records of a group of researchers, who have different mean citation rates (and optionally, different importance coefficients), in such a way to keep the overall citation index as up to date as possible. The updater is subject to a total update rate constraint that it needs to distribute among individual researchers. For this problem, we use a metric similar to the age of information: the *long-term average difference* between the actual citation numbers and the citation numbers according to the latest updates. We show that, in order to minimize this difference metric, the updater should allocate its total update capacity to researchers proportional to the *square roots* of their mean citation rates. That is, more prolific researchers should be updated more often, but there are diminishing returns due to the concavity of the square root function.

In Chapter 10, we consider the problem of real-time timely tracking of multiple independent *binary* random processes again via a Poisson updater with the limited total update rate constraint. As a particular example, we study the problem of timely tracking of infection status (e.g., covid-19) of individuals in a population. Here, a health care provider wants to detect infected people as well as people who recovered from the disease as quickly as possible. In order to measure the timeliness of the tracking process, we use the long-term average difference between the actual infection status of the people and their real-time estimate by the health care provider based on the most recent test results. We first find an analytical expression for this average difference for given test rates, and given infection and recovery rates of people. Then, we propose an alternating minimization based algorithm to minimize this average difference. We observe that if the total test rate is limited, instead of testing all members of the population equally, only a portion of the population is tested based on their infection and recovery rates. We also observe that increasing the total test rate helps track the infection status better. In addition, an increased population size increases diversity of people with different infection and recovery rates, which may be exploited to spend testing capacity more efficiently, thereby improving the system performance. Further, here, depending on the health care provider's preferences, test rate allocation can be altered to detect either the infected people or the recovered people more quickly.

In Chapter 11, we study two closely related problems: anomaly detection in sensor networks and testing for infections in human populations. In both problems, we have many nodes (sensors, humans), and each node exhibits an event of interest (anomaly, infection) with a certain probability. We want to keep track of the anomaly/infection status of all nodes at a central location. We develop a *group updating* scheme, akin to group testing, which updates a central location about the status of each member of the population by appropriately grouping their individual status. Unlike group testing, which uses the expected number of tests as a metric, in group updating, we use the expected age of information at the central location as a metric. We determine the optimal group size to minimize the age of information. We show that, when the probability of anomaly/infection is small, the proposed group updating policy yields smaller age compared to a sequential updating policy.

In Chapter 12, we present conclusions of this dissertation.

CHAPTER 2

Age of Information with Soft Updates

2.1 Introduction

In this chapter, we consider a typical information update system as shown in Fig. 2.1. Starting from time zero, information at the receiver gets stale over time, i.e., the age increases linearly. A time comes when the information source decides to update the information receiver. In the existing literature, this is a *hard* update, which is contained in an information packet. This hard update *takes effect* and reduces the age instantaneously to the age of the packet itself at the time of its arrival at the receiver. This is denoted as *instantaneous decay* in Fig. 2.1. The time for the update to take effect (denoted by c_1 for the first update) is either random [2, 4, 8, 9, 17, 31, 61, 93, 135–139], or fixed and deterministic [50, 51], or zero [37–48]. Essentially, this is the time for the update packet to *travel* from the transmitter to the receiver, and when it arrives, it drops the age instantaneously. This travel time is random if the update goes through a queue, it is fixed if the update goes through a wireless channel with a non-negligible distance between the transmitter and the receiver, and it is zero if the update goes through a channel with a negligible

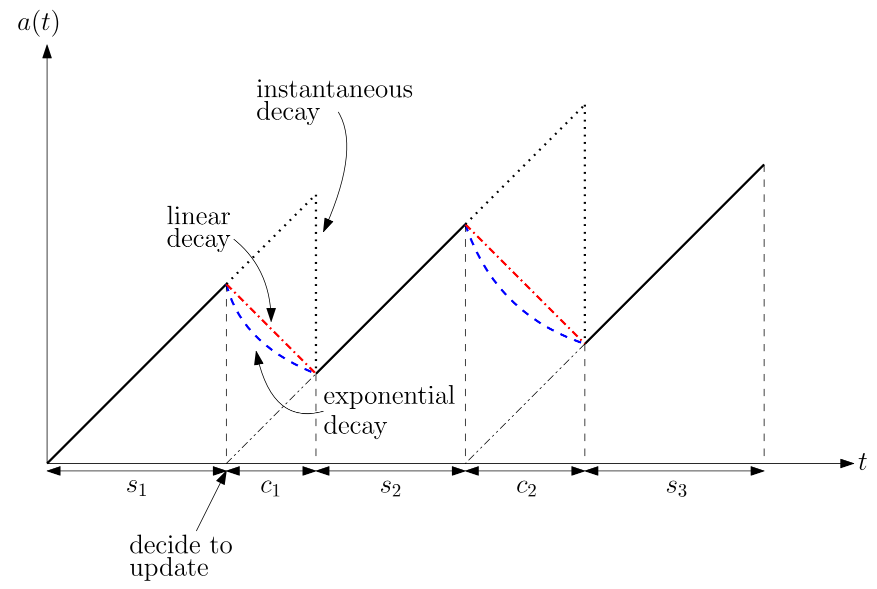


Figure 2.1: Update models: Hard updates (instantaneous decay) and soft updates (exponential and linear decay).

distance. In contrast, in this work, the soft update begins reducing the age at the time of information source making a decision to update. However, the drop in age is not instantaneous, rather it is *gradual* over time.

We consider two models for the soft update process: In the first model, the rate of decrease in age is proportional to the current age; see (2.1). This is motivated by the fact that new information is most valuable when the current information is most aged, i.e., when the new information is most innovative. This model leads to an exponential decay in the age (denoted by *exponential decay* in Fig. 2.1). Note also that, the exponential decay in the age is consistent with information dissemination in human interactions as well as in social media feeds, where the most important information is conveyed/displayed first, reducing the age faster initially, and less important information is conveyed/displayed next, reducing the age slower subsequently. In the second model, the rate of decrease in age is not a function of the current age, rather it is constant; see (2.2). In this case, the age decreases linearly (denoted by *linear decay* in Fig. 2.1).

In this chapter, we determine the optimum updating schemes for *soft update systems*. We are given the total system duration over which the average age is calculated T, the number of update periods (i.e., the number of times information provider and information receiver are allowed to meet) N, and the total allowed update duration T_c. We solve for the optimum start times of the soft updates and their optimum durations in order to minimize the overall age.

We show that for both exponentially and linearly decaying age models, the optimal policy is to have exactly N soft updates, completely utilize the given total update duration T_c, and divide the total update duration T_c equally among N updates. We note that when T_c is large compared to T, we may have multiple optimal solutions. In order to generalize the solution for both models and for any T_c, we choose the optimal policy which allocates equal amount of time for each soft update. For the exponentially decaying age model, if T_c is small compared to T, the optimal policy schedules the updates regularly; if T_c is large enough, the system starts updating at time zero, proceeds to update continually until T_c is completely utilized, and lets age grow then on until the end. For the linearly decaying age model, if T_c is small compared to T, the optimal policy schedules the updates regularly and the age after each soft update goes down exactly to zero; if T_c is large enough, age not only goes down to zero after each soft update, but also stays at zero for some time after each soft update. In addition, for the exponentially decaying age model with small T_c and for the linearly decaying age model for all T_c, we show that the resulting age decreases with N.

2.2 System Model and the Problem

Let $a(t)$ be the instantaneous age at time t. Without loss of generality, let $a(0) = 0$. When there is no update, the age increases linearly with time. We consider two different soft update models. In the first model, the rate of decrease in age is proportional to the current age:

$$\frac{da(t)}{dt} = -\alpha a(t) \tag{2.1}$$

where α is a fixed constant. In this model, the age decreases exponentially during a soft update period. In the second model, the rate of decrease in age does not depend on the current age, instead it remains constant:

$$\frac{da(t)}{dt} = -\alpha \tag{2.2}$$

where α is a fixed constant. In this model, the age decreases linearly during a soft update period.

Let us denote the beginning of the ith soft update period by t_i and the end of the ith soft update period by t'_i. Then, the age evolves as:

$$a(t) \triangleq \begin{cases} a(t'_{i-1}) + t - t'_{i-1}, & t'_{i-1} < t < t_i \\ f(a(t_i), \alpha, t), & t_i < t < t'_i \end{cases} \tag{2.3}$$

where $f(a(t_i),\alpha,t) = a(t_i)e^{-\alpha(t-t_i)}$ for the exponentially decaying age model, and $f(a(t_i),\alpha,t) = (a(t_i)-\alpha(t-t_i))^+$ for the linearly decaying age model, where $(x)^+ = x$ for $x > 0$ and $(x)^+ = 0$ for $x \leq 0$. For both models, if the current age is larger than zero, age decreases during an update period. For the linearly decaying age model, depending on the update duration and the age at the beginning of the update, the age can go down to zero. Here, we consider the most general case where the age can stay at zero if the duration of the update period is large enough.[1] For the exponentially decaying age model, age stays at zero only if we have an update starting at time $t = 0$. Otherwise, age never goes down to zero in a finite update duration.

Our objective is to minimize the average age of information (AoI) of the system subject to a total of N soft update periods, a total update duration of T_c, over a total session duration of T. We formulate the problem as:

$$\begin{aligned} \min_{\{t_i,t_i'\}} \quad & \frac{1}{T}\int_0^T a(t)dt \\ \text{s.t.} \quad & \sum_{i=1}^{N}(t_i'-t_i) \leq T_c \end{aligned} \tag{2.4}$$

We define the duration of the ith update period as $c_i = t_i' - t_i$, and the ith aging period as $s_i = t_i - t_{i-1}'$. For convention, we let $t_0' = 0$, and $t_{N+1} = T$. Additionally, we denote the age at the beginning of the ith soft update period by x_i, and the age at the end of the ith soft update period by y_i. Therefore, we obtain three equivalent

[1]In [62], for the linearly decaying age model, we consider the case where we terminate an update process if the current age goes down to zero. In this chapter, we assume that an update process can continue after the current age becomes zero. During this period, since the update process is on, the system does not age, i.e., the age stays at zero.

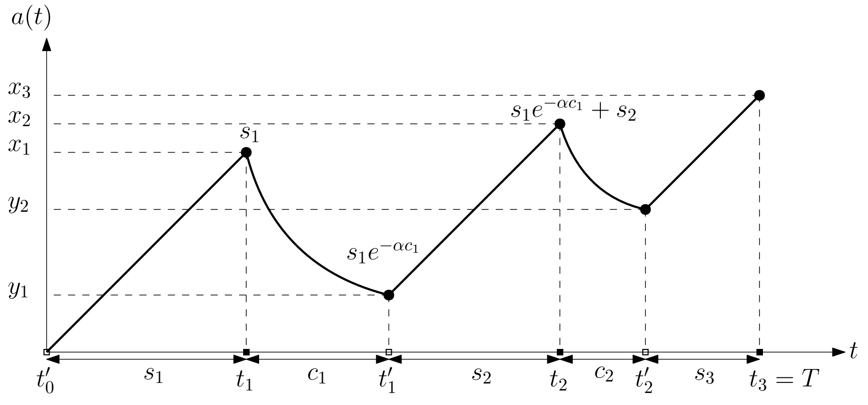


Figure 2.2: A general example evolution of age in the case of exponentially decaying age.

sets of variables to describe the system: $\{t_i, t'_i\}_{i=1}^N$, $\{s_i, c_i\}_{i=1}^N$, and $\{x_i, y_i\}_{i=1}^N$. We retain these three sets of equivalent variables throughout this chapter; we find it more convenient to express A_T in terms of x_i and y_i for the exponentially decreasing age model, and in terms of s_i and c_i for the linearly decreasing age model. The relationship between (t_i, t'_i), (s_i, c_i), and (x_i, y_i) is shown in Fig. 2.2.

Let $A_T \triangleq \int_0^T a(t)dt$ be the total age. Note that minimizing $\frac{A_T}{T}$ is equivalent to minimizing A_T since T is a known constant. In the following sections, we provide the optimal policies that minimize the age for the cases of exponentially and linearly decaying age models.

2.3 Exponentially Decaying Age Model

In the case of exponentially decaying age, the age function evolves as in Fig. 2.2. Age, in this case, is given in terms of x_i and y_i as:

$$A_T = \sum_{i=1}^{N} \frac{x_i^2}{2} - \frac{y_i^2}{2} + \frac{1}{\alpha}(x_i - y_i) + \frac{x_{N+1}^2}{2} \tag{2.5}$$

We minimize A_T in (2.5) by choosing x_i and y_i, equivalently, by choosing t_i and t'_i, and s_i and c_i, for all i. In the following lemma, we show that the total update time, T_c, should be fully utilized.

Lemma 2.1 *For the exponentially decaying age model, in the optimal policy, we must have* $\sum_{i=1}^{N} c_i = T_c$.

Proof: We prove this by contradiction. Assume that there exists an optimal policy such that $\sum_{i=1}^{N} c_i < T_c$. Then, we can simply obtain another feasible policy by increasing one of the c_i and decreasing one of the s_j. Note that this new policy yields a smaller age. Thus, we reached a contradiction, and $\sum_{i=1}^{N} c_i = T_c$ must be satisfied. ■

Thus, from Lemma 2.1, the total update time should be fully used. Then, we need to determine when to start a soft update and the duration of each soft update. In the case of $T_c = T$, the optimal policy is to start updating at $t = 0$ and continue to update until $t = T$. The optimal age in this case is $A_T = 0$. Thus, for the rest of this section, we consider the case where $T_c < T$. We formulate the optimization problem as:

$$\begin{aligned}
\min_{\{x_i, y_i\}} \quad & \sum_{i=1}^{N} \frac{x_i^2}{2} - \frac{y_i^2}{2} + \frac{1}{\alpha}(x_i - y_i) + \frac{x_{N+1}^2}{2} \\
\text{s.t.} \quad & \sum_{i=1}^{N} \frac{1}{\alpha} \log\left(\frac{x_i}{y_i}\right) \leq T_c \\
& \sum_{i=1}^{N} \left(x_i - y_i + \frac{1}{\alpha} \log\left(\frac{x_i}{y_i}\right)\right) + x_{N+1} = T \\
& y_i \leq x_i, \quad y_i \leq x_{i+1}, \quad x_i \geq 0, \quad y_i \geq 0
\end{aligned} \tag{2.6}$$

where the cost function is the age expression in (2.5); the first constraint is the constraint on the total soft update duration which is obtained by noting that the ith update duration c_i is expressed in terms of x_i and y_i as $y_i = x_i e^{-\alpha c_i}$ therefore, $c_i = \frac{1}{\alpha}\log\left(\frac{x_i}{y_i}\right)$; the second constraint is the total session duration constraint which is the sum of aging durations s_i and update durations c_i where s_i is given in terms of x_i and y_i as $s_i = x_i - y_{i-1}$ with the convention of $y_0 = 0$; and the third (last) set of constraints state that age in the update period decreases ($y_i \leq x_i$), age in the aging period increases ($y_i \leq x_{i+1}$), and age at all times is non-negative ($x_i \geq 0$, $y_i \geq 0$).

We write the Lagrangian for the problem in (2.6) as:

$$\begin{aligned}\mathcal{L} =& \sum_{i=1}^{N} \frac{x_i^2}{2} - \frac{y_i^2}{2} + \frac{1}{\alpha}(x_i - y_i) + \frac{x_{N+1}^2}{2} + \lambda\left(\sum_{i=1}^{N} \frac{1}{\alpha}\log\left(\frac{x_i}{y_i}\right) - T_c\right) \\ &+ \beta\left(T - \left(\sum_{i=1}^{N}\left(x_i - y_i + \frac{1}{\alpha}\log\left(\frac{x_i}{y_i}\right)\right) + x_{N+1}\right)\right) \\ &+ \sum_{i=1}^{N}\gamma_i(y_i - x_i) + \sum_{i=1}^{N}\theta_i(y_i - x_{i+1}) - \sum_{i=1}^{N+1}\mu_i x_i - \sum_{i=1}^{N}\nu_i y_i \end{aligned} \tag{2.7}$$

where $\lambda \geq 0$, $\gamma_i \geq 0$, $\theta_i \geq 0$, $\mu_i \geq 0$, $\nu_i \geq 0$, and β can be anything. Note that the problem given in (2.6) is not convex. Thus, KKT conditions are necessary but not sufficient for the optimal solution. The KKT conditions are:

$$\frac{\partial \mathcal{L}}{\partial x_1} = x_1 + \frac{1}{\alpha} + \frac{\lambda}{\alpha x_1} - \beta\left(1 + \frac{1}{\alpha x_1}\right) - \gamma_1 - \mu_1 = 0 \tag{2.8}$$

$$\frac{\partial \mathcal{L}}{\partial x_i} = x_i + \frac{1}{\alpha} + \frac{\lambda}{\alpha x_i} - \beta\left(1 + \frac{1}{\alpha x_i}\right) - \gamma_i - \theta_{i-1} - \mu_i = 0, \quad i = 2, \ldots, N \tag{2.9}$$

$$\frac{\partial \mathcal{L}}{\partial x_{N+1}} = x_{N+1} - \beta - \theta_N - \mu_{N+1} = 0 \tag{2.10}$$

$$\frac{\partial \mathcal{L}}{\partial y_i} = -\, y_i - \frac{1}{\alpha} - \frac{\lambda}{\alpha y_i} + \beta\left(1 + \frac{1}{\alpha y_i}\right) + \gamma_i + \theta_i - \nu_i = 0, \quad i = 1, \ldots, N \tag{2.11}$$

The complementary slackness conditions are:

$$\lambda\left(\sum_{i=1}^{N} \frac{1}{\alpha}\log\left(\frac{x_i}{y_i}\right) - T_c\right) = 0 \tag{2.12}$$

$$\beta\left(T - \left(\sum_{i=1}^{N}\left(x_i - y_i + \frac{1}{\alpha}\log\left(\frac{x_i}{y_i}\right)\right) + x_{N+1}\right)\right) = 0 \tag{2.13}$$

$$\gamma_i(y_i - x_i) = 0 \tag{2.14}$$

$$\theta_i(y_i - x_{i+1}) = 0 \tag{2.15}$$

$$\mu_i x_i = 0 \tag{2.16}$$

$$\nu_i y_i = 0 \tag{2.17}$$

In the following, we consider two cases separately: $x_1 > 0$ and $x_1 = 0$ in the optimal solution. First, we investigate the case when $x_1 > 0$.

2.3.1 The Optimal Solution Structure When $x_1 > 0$

Since $x_1 > 0$, from the complementary slackness conditions, we have $\mu_1 = 0$. Since $y_1 = x_1 e^{-\alpha c_1}$, we have $y_1 > 0$. Due to $x_2 \geq y_1$, we have $x_2 > 0$. Continuing similarly, we have $y_i > 0$ and $x_i > 0$ for all i. Thus, $\mu_i = 0$ and $\nu_i = 0$ for all i. In addition, due to Lemma 2.1, there exists at least one i such that $x_i > y_i$. For these cases, $\gamma_i = 0$. Since $T > T_c$, we have at least one j such that $x_{j+1} > y_j$ and corresponding $\theta_j = 0$. Then, we have four possible cases. Next, we investigate them separately.

2.3.1.1 Case A: $x_i > y_i$ and $x_{i+1} > y_i$ for all i

In this case, we have N strict updating and correspondingly $N+1$ strict aging periods. This case is shown in Fig. 2.3(a). Since $x_i > y_i$ and $x_{i+1} > y_i$ for all i, from the complementary slackness conditions, we have $\gamma_i = 0$ and $\theta_i = 0$ for all i. Thus, (2.8)-(2.11) become:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_i} =& x_i + \frac{1}{\alpha} + \frac{\lambda}{\alpha x_i} - \beta\left(1 + \frac{1}{\alpha x_i}\right) = 0, \quad i = 1, \dots, N && (2.18)\\ \frac{\partial \mathcal{L}}{\partial x_{N+1}} =& x_{N+1} - \beta = 0 && (2.19)\\ \frac{\partial \mathcal{L}}{\partial y_i} =& -y_i - \frac{1}{\alpha} - \frac{\lambda}{\alpha y_i} + \beta\left(1 + \frac{1}{\alpha y_i}\right) = 0, \quad i = 1, \dots, N && (2.20)\end{aligned}$$

Note that the right hand sides of $\frac{\partial \mathcal{L}}{\partial x_i}$ and $\frac{\partial \mathcal{L}}{\partial y_i}$ in (2.18) and (2.20) are the same second degree equalities. Since we consider the case where $x_i > y_i$ for all i, the larger root of this equality gives x_i and the smaller root gives y_i. Rewriting (2.18) in terms of a single variable z, we have,

$$z + \frac{1}{\alpha} + \frac{\lambda}{\alpha z} - \beta\left(1 + \frac{1}{\alpha z}\right) = 0 \tag{2.21}$$

which is equivalent to,

$$\alpha z^2 + z(1 - \beta\alpha) + (\lambda - \beta) = 0 \tag{2.22}$$

The roots of this equation are,

$$z_1 = \frac{-(1-\beta\alpha) + \sqrt{(1-\alpha\beta)^2 - 4\alpha(\lambda-\beta)}}{2\alpha} \tag{2.23}$$

$$z_2 = \frac{-(1-\beta\alpha) - \sqrt{(1-\alpha\beta)^2 - 4\alpha(\lambda-\beta)}}{2\alpha} \tag{2.24}$$

and we have $x_i = z_1$ and $y_i = z_2$, for all i. Note that in order to have two positive roots, we need $1 - \beta\alpha < 0$. Thus, we have:

$$x_{N+1} = \beta > \frac{1}{\alpha} \tag{2.25}$$

where we also used (2.19). Since $c_i = \frac{1}{\alpha}\log\left(\frac{x_i}{y_i}\right)$ and $\sum_{i=1}^{N} c_i = T_c$ and since all x_i are equal among themselves and all y_i are equal among themselves, we have all c_i equal and $c_i = \frac{T_c}{N}$.

Next, we note from (2.22) that,

$$x_{N+1} = x_i + y_i + \frac{1}{\alpha} \tag{2.26}$$

Further, by using (2.12), (2.13) and Lemma 2.1, we obtain,

$$N(x_i - y_i) + x_{N+1} = T - T_c \tag{2.27}$$

Substituting (2.26) into (2.27), and noting that $\frac{1}{\alpha}\log\left(\frac{x_i}{y_i}\right) = c_i = \frac{T_c}{N}$, we solve for

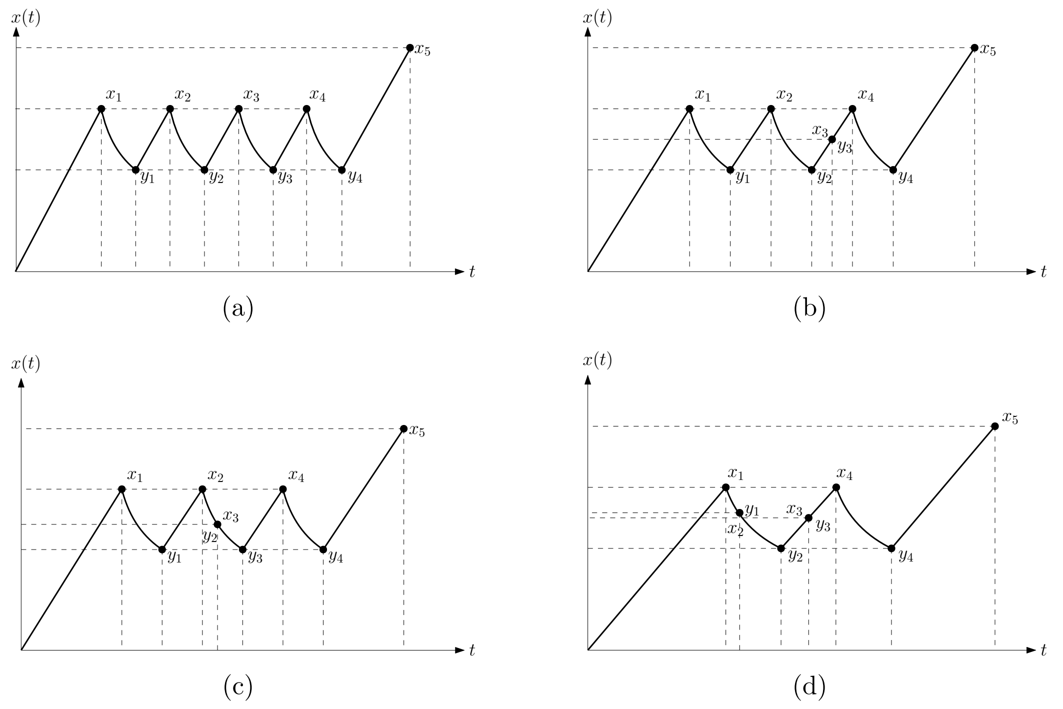


Figure 2.3: Depiction of the cases for the exponentially decaying age model with $x_1 > 0$ (a) where $x_i > y_i$ and $x_{j+1} > y_j$ for all i and j, (b) where $x_i = y_i$ for some i, (c) where $x_{j+1} = y_j$ for some j, (d) where $x_i = y_i$ and $x_{j+1} = y_j$ for some i and j.

x_i as,

$$x_i = \frac{T - T_c - \frac{1}{\alpha}}{(N+1) - (N-1)e^{-\frac{\alpha T_c}{N}}}, \quad i = 1, \ldots, N \tag{2.28}$$

and

$$x_{N+1} = \frac{(T - T_c)\left(1 + e^{-\frac{\alpha T_c}{N}}\right) + \frac{N}{\alpha}}{(N+1) - (N-1)e^{-\frac{\alpha T_c}{N}}} \tag{2.29}$$

With this solution, the minimum age, A_T, is:

$$A_T = \frac{1}{2}\left(T - T_c - \frac{1}{\alpha}\right)^2 \frac{1 + e^{-\frac{\alpha T_c}{N}}}{(N+1) - (N-1)e^{-\frac{\alpha T_c}{N}}} + \frac{1}{\alpha}(T - T_c) - \frac{1}{2\alpha^2} \tag{2.30}$$

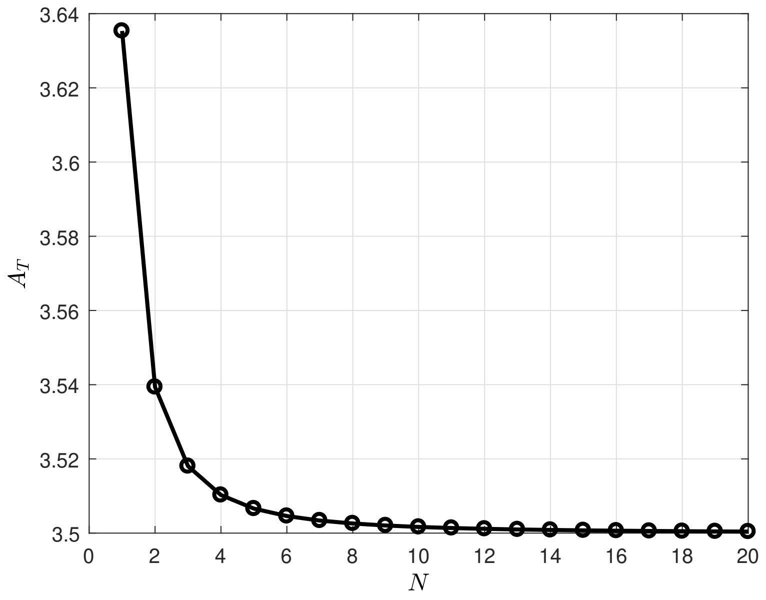


Figure 2.4: Minimum age as a function of N in the exponentially decaying age case for $T = 5$, $T_c = 2$, and $\alpha = 1$.

We note that A_T is monotonically decreasing with respect to N in *Case A*. To see this, we note that the derivative of A_T in (2.30) with respect to N is equal to,

$$\frac{\partial A_T}{\partial N} = C \frac{2\left(\frac{\alpha T_c}{N}\right) e^{-\frac{\alpha T_c}{N}} + e^{-\frac{2\alpha T_c}{N}} - 1}{\left((N+1) - (N-1)e^{-\frac{\alpha T_c}{N}}\right)^2} \tag{2.31}$$

where $C = \frac{1}{2}\left(T - T_c - \frac{1}{\alpha}\right)^2$. Note that C and the denominator in (2.31) are always positive.

Next, letting $a = \frac{\alpha T_c}{N}$, the numerator of (2.31) becomes $2ae^{-a}\left(1 - \frac{\sinh(a)}{a}\right)$. Since $\sinh(a) \geq a$ for all $a \geq 0$, and therefore, $\frac{\sinh(a)}{a} \geq 1$ for all $a \geq 0$, this implies that the numerator of (2.31) is always negative, implying that $\frac{dA_T}{dN} \leq 0$. As an aside, we plot A_T versus N for $T = 5$, $T_c = 2$, and $\alpha = 1$ in Fig. 2.4. Note that A_T is a decreasing function with respect to N with a limit:

$$\lim_{N\to\infty} A_T = \frac{1}{2}\left(T - T_c - \frac{1}{\alpha}\right)^2 \frac{2}{2+\alpha T_c} + \frac{1}{\alpha}(T - T_c) - \frac{1}{2\alpha^2} \tag{2.32}$$

2.3.1.2 Case B: $x_i = y_i$ for some i and $x_{j+1} > y_j$ for all j

This case is shown in Fig. 2.3(b). This is equivalent to *Case A* with $N' = N - n$, where n is the total number of update processes with $x_i = y_i$. We know from *Case A* that A_T decreases with N. Thus, *Case B* cannot be optimal.

2.3.1.3 Case C: $x_i > y_i$ for all i and $x_{j+1} = y_j$ for some j

This case is shown in Fig. 2.3(c). Similar to *Case B*, this case is equivalent to *Case A* with $N' = N - m$, where m is the total number of aging processes with $x_{j+1} = y_j$. Thus, *Case C* cannot be optimal.

2.3.1.4 Case D: $x_i = y_i$ for some i and $x_{j+1} = y_j$ for some j

This case is shown in Fig. 2.3(d). This is equivalent to *Case A* with $N' = N - k$, where k is the total number of update and aging processes with $x_i = y_i$ and $x_{j+1} = y_j$ subtracting $i = j$ cases. Thus, *Case D* cannot be optimal.

Thus, we see that if we have $x_1 > 0$, the optimal solution only comes from *Case A*. In addition, from (2.25) and (2.27), in order to have $x_1 > 0$, we need:

$$\frac{1}{\alpha} < x_{N+1} < T - T_c \tag{2.33}$$

Therefore, in the optimal solution, if we have $x_1 > 0$, then $T > T_c + \frac{1}{\alpha}$ should be satisfied.

As a result, if $x_1 > 0$ in the optimal solution, this should happen for T and T_c that satisfy $T > T_c + \frac{1}{\alpha}$, i.e., T_c is relatively small in relation to T, and in this case, the optimal solution is to update N times with equal update durations, i.e., $c_i = \frac{T_c}{N}$, for all i as shown in Fig. 2.3(a).

Next, we study the optimal solution structure when $x_1 = 0$.

2.3.2 The Optimal Solution Structure When $x_1 = 0$

So far, we studied the optimal solution structure when $x_1 > 0$. We see that this case requires $T > T_c + \frac{1}{\alpha}$. Thus, when $T \leq T_c + \frac{1}{\alpha}$, we have $x_1 = 0$. Since $y_1 = x_1 e^{-\alpha c_1}$, we have $y_1 = 0$. In the following, we show that if $T \leq T_c + \frac{1}{\alpha}$, then the optimal policy is to keep the age equal to zero starting from $t = 0$ till $t = T_c$, and let the age grow from $t = T_c$ till $t = T$.

We see that if $T \leq T_c + \frac{1}{\alpha}$, then $x_1 = y_1 = 0$. Also, $0 \leq c_1 \leq T_c$. If $c_1 = T_c$, then the optimal policy is exactly as descibed above, i.e., start the update policy at $t = 0$ and continue updating until $t = c_1 = T_c$, and stop updating then, i.e., let the age grow until $t = T$. If $c_1 < T_c$, we need to first show that $x_2 = 0$, and therefore, $y_2 = 0$. We prove this by contradiction. Assume that there exists an optimal policy such that $T \leq T_c + \frac{1}{\alpha}$, $x_1 = y_1 = 0$, and $x_2 > 0$. Since the age stays at zero during c_1, we can formulate a new age minimization problem starting from $t = c_1$. For the new problem, $T' = T - c_1$, $T_c' = T_c - c_1$, and $N' = N - 1$. Since

$T' = T - c_1 \leq T_c - c_1 + \frac{1}{\alpha} = T_c' + \frac{1}{\alpha}$, we have $T' \leq T_c' + \frac{1}{\alpha}$. Thus, for the new problem, we reach a contradiction and we must have $x_2 = 0$ as well as $y_2 = 0$. At this point, we have $0 \leq c_2 \leq T_c - c_1$. If $c_2 = T_c - c_1$, we have the desired policy described above. If not, we repeat the same steps to argue that $x_3 = 0$, and thus, $y_3 = 0$. Then, we select $c_3 \in [0, T_c - c_1 - c_2]$. Thus, for the remaining terms, we can either argue that $c_i = T_c - \sum_{j=1}^{i-1} c_j$ or show that $x_{i+1} = y_{i+1} = 0$ and select c_{i+1} accordingly. At the end, the optimal policy is to update starting from $t = 0$, proceed to update continually until $t = T_c$, and then let the age grow until T.

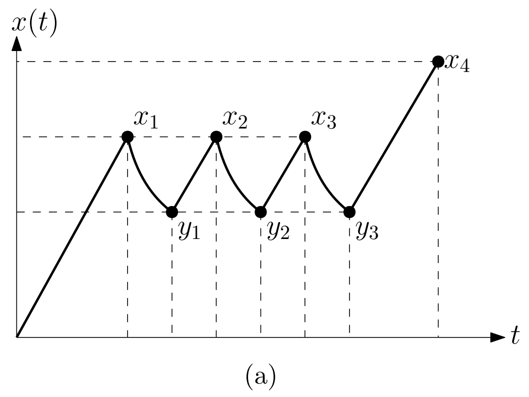
Here, we may view the optimal solution in multiple ways: We may view it as a single update that lasts $c_1 = T_c$ second, or we may view it N updates that altogether last $c_1 + \cdots + c_N = T_c$ seconds, or N' updates where $1 < N' < N$ with appropriate selection of corresponding c_i to sum up to T_c. Even though we have such multiple optimal solutions, we choose the one with N updates with equal update durations (to be consistent with the solution in the previous sub-section), i.e., $c_i = \frac{T_c}{N}$, for all i. Thus, we have $x_i = y_i = 0$ for $i = 1, \ldots, N$ and $x_{N+1} = T - T_c$.

With this solution, the minimum age, A_T, is:

$$A_T = \frac{(T - T_c)^2}{2} \tag{2.34}$$

We note that A_T in (2.34) does not decrease with N unlike A_T in (2.30).

Finally, we summarize the optimal policy for the exponentially decaying age case combining the results in Sub-sections 2.3.1 and 2.3.2. If $T_c < T - \frac{1}{\alpha}$, i.e., the allowed update duration is relatively small with respect to the total session duration,



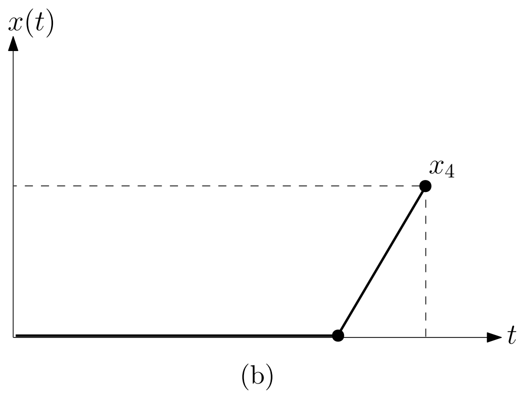


Figure 2.5: Optimal solution for the exponentially decaying age case: (a) When $T_c < T - \frac{1}{\alpha}$ (relatively small update duration). (b) When $T_c > T - \frac{1}{\alpha}$ (relatively large update duration).

then the optimal policy is to update N times with equal update durations $c_i = \frac{T_c}{N}$. Also, in this case, all x_i for $i = 1, \dots, N$ should be equal as given in (2.28), and all y_i for $i = 1, \dots, N$ should be equal as well. An example age evolution curve for this case for $N = 3$ is shown in Fig. 2.5(a). If $T_c > T - \frac{1}{\alpha}$, i.e., the allowed update duration is relatively large compared to the total session duration, then the optimal policy is to update starting from $t = 0$ till $t = T_c$, and then let the age grow afterwards until $t = T$. There are multiple optimal assignments of total update duration T_c to c_i in this case; we choose $c_i = \frac{T_c}{N}$ again for symmetry with the previous case. Also, in this case, all x_i for $i = 1, \dots, N$ are equal and equal to zero, and all y_i for $i = 1, \dots, N$ are equal and equal to zero as well. An example age evolution curve for this case is shown in Fig. 2.5(b).

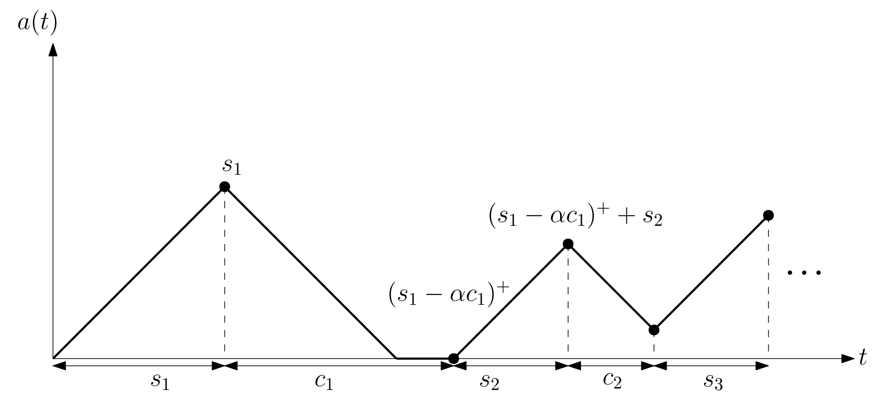


Figure 2.6: A general example evolution of age in the case of linearly decaying age.

2.4 Linearly Decaying Age Model

In this section, we consider the linearly decaying age model where the aging process can be slower or faster than the updating process. We consider the most general case by allowing the slope in the soft update policy, α, to be arbitrary. In additional, when the duration of soft update process is sufficiently large, the instantaneous age can be reduced to zero. In this case, we can further continue the soft update process, and as a result, keep the age at the level of zero, i.e., not allow it to grow. A general example evolution of age for the linearly decaying age model is shown in Fig. 2.6. Age, in this case, is given as:

$$\begin{aligned} A_T =& \frac{\alpha+1}{2\alpha}\sum_{i=1}^{N}\left(\left(s_i+\sum_{j=0}^{i-1}(s_j-\alpha c_j)^+\right)^2-\left(\sum_{j=1}^{i}(s_j-\alpha c_j)^+\right)^2\right) \\ &+\frac{(s_{N+1}+\sum_{j=1}^{N}(s_j-\alpha c_j)^+)^2}{2} \end{aligned} \tag{2.35}$$

where $c_0=0$, $s_0=0$, and $s_{N+1}=T-\sum_{i=1}^{N}(s_i+c_i)$.

Next, we identify some important properties of the optimal solution. First, the following lemma states that, in the optimal solution, total update time, T_c, should be completely utilized.

Lemma 2.2 *For the linearly decaying age model, in the optimal policy, we must have* $\sum_{i=1}^{N} c_i = T_c$.

Proof: We prove this by contradiction. Assume that in the optimal policy, we have $\sum_{i=1}^{N} c_i < T_c$. First, let us choose the smallest index, j, such that $a(t'_j) > 0$. We can decrease the age further by increasing c_j. This policy is still feasible since the total update time constraint is not tight. Thus, we continue to increase c_j until either $a(t'_j) = 0$ or $\sum_{i=1}^{N} c_i = T_c$. If $a(t'_j) = 0$ and $\sum_{i=1}^{N} c_i < T_c$, we move to the second smallest index such that the age at the end of the update period is not zero and apply the same procedure. We apply this procedure until $a(t'_i) = 0$ for all i. At the end, if we obtain $\sum_{i=1}^{N} c_i < T_c$ and $a(t'_i) = 0$ for all i, we can further decrease the age by increasing the duration of any update process by the amount $T_c - \sum_{i=1}^{N} c_i$. Since $a(t'_i) = 0$ for all i, the age will stay at zero. Thus, we obtain a new policy where $\sum_{i=1}^{N} c_i = T_c$. This new policy has smaller age at each step, implying we have reached a contradiction. Thus, in the optimal policy, $\sum_{i=1}^{N} c_i = T_c$. ■

From Lemma 2.2, we see that the total update time, T_c, should be fully used. Thus, when $T_c = T$, the optimal solution is to update the system starting from $t = 0$ to $t = T$. The optimal age in this case is $A_T = 0$. When $T_c < T$, we have time intervals where the system ages. If we decrease T_c, the total time where the age stays at zero decreases since there will be no update for $T - T_c$ and some portion of an update process can be used to decrease the age to zero. Let us first consider the case where $\sum_{i=1}^{k}(s_i - \alpha c_i) \geq 0$, for all $k = 1, \dots, N$. In other words, we consider the case where each soft update process ends before or as soon as instantaneous age reaches zero. After providing a solution for this specific case, we generalize the solution to the most general case where the age can stay at zero. Thus, we formulate the problem with this condition enforced, as follows:

$$\begin{aligned} \min_{\{s_i, c_i\}} \quad & A_T \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} s_i + \sum_{i=1}^{N} c_i = T \\ & \sum_{i=1}^{N} c_i \leq T_c \\ & \sum_{i=1}^{k} s_i - \alpha c_i \geq 0, \quad \forall k \end{aligned} \tag{2.36}$$

where A_T in the cost function is the age expression in (2.35); the first constraint is the total session duration constraint which is the sum of aging and update durations; the second constraint is the constraint on the total soft update duration; and the third (last) constraint enforces that each update duration ends before or as soon as the age goes down to zero as discussed above.

This is not a convex optimization problem as the objective function is not convex. Our approach will be to lower bound the objective function, minimize this lower bound, and then show that this minimized lower bound can be achieved with a certain feasible selection of the variables. First, the following lemma states that, in the optimal solution, the age should be equal to zero at the end of each and every soft update period, i.e., the update period should never end before the age goes down exactly to zero.

Lemma 2.3 *For the linearly decaying age model, for the problem in (2.36) which terminates updates if the age reaches zero, in the optimal policy, the age should be exactly equal to zero at the end of each soft update period i.e.,* $a(t_i') = 0$ *for all* i*. In addition,* $c_i = \frac{T_c}{N}$*,* $s_i = \frac{\alpha T_c}{N}$ *for all* $i = 1, \ldots, N$*, and* $s_{N+1} = T - (\alpha+1)T_c$*.*

Proof: We first note that A_T in (2.35) can equivalently be written as:

$$A_T = \frac{\alpha+1}{2}\left(\alpha\sum_{i=1}^{N} c_i^2 + 2\sum_{i=1}^{N}(s_i - \alpha c_i)\left(\sum_{j=i}^{N} c_j\right)\right) + \frac{(T-(\alpha+1)T_c)^2}{2} \tag{2.37}$$

We next note that, even though we do not know the sign of each $(s_i - \alpha c_i)$ in (2.37) at this point, we know that the entirety of the middle term in (2.37) is always non-negative since:

$$\sum_{i=1}^{N}(s_i - \alpha c_i)\left(\sum_{j=i}^{N} c_j\right) = \sum_{i=1}^{N}\left(\sum_{j=1}^{i} s_j - \alpha c_j\right) c_i \tag{2.38}$$

where the right hand side is non-negative due to the constraints in (2.36). Thus, we lower bound (2.37) by setting the middle term as zero by choosing $s_i = \alpha c_i$ for all i which also implies that the age is equal to zero at the end of each soft update period. Then, minimizing the lower bound becomes equivalent to minimizing $\sum_{n=1}^{N} c_i^2$ subject to $\sum_{i=1}^{N} c_i = T_c$, whose solution is $c_i = \frac{T_c}{N}$. Then, we can choose $s_i = \alpha c_i$ and $c_i = \frac{T_c}{N}$ for all $i = 1, \ldots, N$, and $s_{N+1} = T - (\alpha+1)T_c$. ■

Next, we extend our solution to include the cases where the age can stay as zero. Towards that end, in the following lemma, we prove that the age cannot stay at zero for some update process(es) unless age becomes zero at the end of each and every update.

Lemma 2.4 *For the linearly decaying age model, in the optimal policy, if the age stays at zero for some update process(es), then the age should be equal to zero after each update period.*

Proof: We prove this by contradiction. Assume that we have an optimal update policy where the age stays at zero for a total of T_0 amount of time and yet there exists an update period i where $s_i - \alpha c_i > 0$, i.e., the age does not go down to zero after the ith update period. Then, subtract T_0 from the total update duration T_c, and consider the age minimization problem with a total update duration of $T_c' = T_c - T_0$. We know from Lemma 2.3 that if the age does not decrease down to zero after each update, the update policy cannot be optimal. Therefore, there exists a policy which yields a smaller age than the assumed optimal update policy. Thus, we have reached a contradiction and the original update policy cannot be optimal. Hence, if the age stays at zero for some update process(es), then the age should be equal to zero after each update. ■

Next, we find the optimal solution structure for the case where the age stays at zero for some update process(es).

Lemma 2.5 *For the linearly decaying age model, in the optimal policy, if the age stays at zero for some update process(es), then the optimal policy is to choose $c_i = \frac{T_c}{N}$ and $s_i = \frac{(T-T_c)\alpha}{\alpha(N+1)+1}$ for $i = 1, \ldots, N$, and $s_{N+1} = \frac{(T-T_c)(\alpha+1)}{\alpha(N+1)+1}$. In addition, we must have $T_c \geq \frac{NT}{(\alpha+1)(N+1)}$.*

Proof: Since we consider the case where the age stays at zero, age at the end of each update process should be equal to zero due to Lemma 2.4. Thus, A_T in (2.35) becomes:

$$A_T = \frac{\alpha+1}{2\alpha}\sum_{i=1}^{N} s_i^2 + \frac{s_{N+1}^2}{2} \tag{2.39}$$

For this case, we need to solve the following problem:

$$\begin{aligned} \min_{\{s_i,c_i\}} \quad & \frac{\alpha+1}{2\alpha}\sum_{i=1}^{N} s_i^2 + \frac{s_{N+1}^2}{2} \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} s_i = T - T_c \\ & s_i - \alpha c_i \leq 0, \quad \forall i \end{aligned} \tag{2.40}$$

The last constraint in (2.40) makes sure that age goes down to zero after each soft update period. We solve this problem using a Lagrangian:

$$\mathcal{L} = \frac{\alpha+1}{2\alpha}\sum_{i=1}^{N} s_i^2 + \frac{s_{N+1}^2}{2} - \lambda\left(\sum_{i=1}^{N+1} s_i - T + T_c\right) \tag{2.41}$$

Taking the derivative with respect to s_i and equating to zero, we obtain $s_i = \frac{\alpha\lambda}{\alpha+1}$ for $i = 1, \dots, N$, and $s_{N+1} = \lambda$. Since $\sum_{i=1}^{N+1} s_i = T - T_c$, the optimal solution is $s_i = \frac{(T-T_c)\alpha}{\alpha(N+1)+1}$ for $i = 1, \dots, N$, and $s_{N+1} = \frac{(T-T_c)(\alpha+1)}{\alpha(N+1)+1}$. Due to the last constraint, we must have $s_i = \frac{(T-T_c)\alpha}{\alpha(N+1)+1} \leq \alpha c_i$. Even though these constraints are satisfied by multiple sets of c_i's, we choose the one with $c_i = \frac{T_c}{N}$. Finally, we need $T_c \geq \frac{NT}{(\alpha+1)(N+1)}$ in order to have feasible selections of $s_i \leq \alpha c_i$ for all i. ■

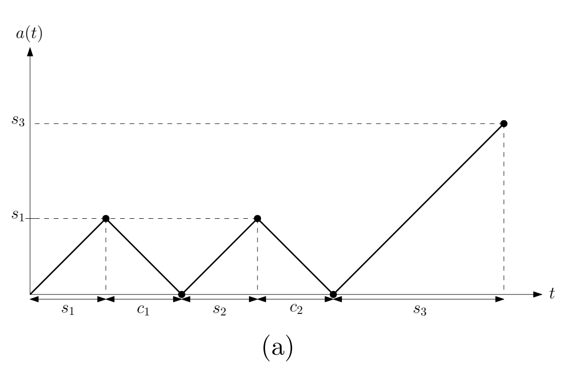


Figure 2.7: Optimal policy structure for the linearly decaying age case: (a) When $T_c < \frac{NT}{(\alpha+1)(N+1)}$ and $\alpha = 1$. (b) When $T_c \geq \frac{NT}{(\alpha+1)(N+1)}$ and $\alpha = 1$.

Finally, we summarize the optimal policy for the linearly decaying age case. If $T_c < \frac{NT}{(\alpha+1)(N+1)}$, i.e., the allowed update duration is relatively small with respect to the total session duration, we are in Lemma 2.3 and the optimal policy is to choose $s_i = \alpha c_i$ and $c_i = \frac{T_c}{N}$ for $i = 1, \dots, N$, and $s_{N+1} = T - (\alpha + 1)T_c$. An example age evolution curve for this case for $N = 2$ is shown in Fig. 2.7(a). If $T_c \geq \frac{NT}{(\alpha+1)(N+1)}$, i.e., the allowed update duration is relatively large compared to the total session duration, we are in Lemma 2.5 and the optimal policy is to choose $s_i = \frac{(T-T_c)\alpha}{\alpha(N+1)+1}$, $c_i = \frac{T_c}{N}$ for $i = 1, \dots, N$, and $s_{N+1} = \frac{(T-T_c)(\alpha+1)}{\alpha(N+1)+1}$.[2] An example age evolution curve for this case for $N = 2$ is shown in Fig. 2.7(b). The optimal policy is to update exactly N times in both cases with the age going down exactly to zero after each update. In addition, if the total update duration T_c is large compared to the total time T then the age stays at zero for some time for all update periods. Finally, we note that the case of age not going down to zero after the second update in the example general age evolution curve shown in Fig. 2.6 can never happen.

[2]In [62, Section IV.B], the same result for $\alpha = 1$ should hold. Therefore, when $T_c < \frac{NT}{2N+2}$, the solution remains the same as in [62, Lemma 3]. When $T_c \geq \frac{NT}{2N+2}$, the optimal solution is to choose $c_i = \frac{T_c}{N}$ and $s_i = \frac{T-T_c}{N+2}$ for $i = 1, \dots, N$, and $s_{N+1} = \frac{2(T-T_c)}{N+2}$.

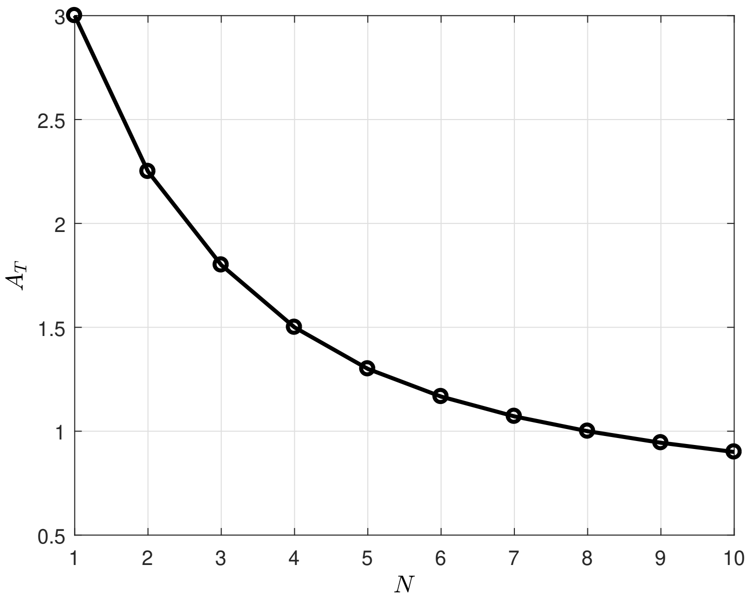


Figure 2.8: Minimum age as a function of N in the linearly decaying age case for $T = 5$, $T_c = 2$ $\alpha = 1$.

Next, we investigate how the final minimum age expression varies as a function of the number of soft update opportunities N. If $T_c < \frac{NT}{(\alpha+1)(N+1)}$, the minimum age is:

$$A_T = \frac{T_c^2}{N}\frac{\alpha(\alpha+1)}{2} + \frac{(T-(\alpha+1)T_c)^2}{2} \tag{2.42}$$

and if $T_c \geq \frac{NT}{(\alpha+1)(N+1)}$, the minimum age is:

$$A_T = \frac{(\alpha+1)(T-T_c)^2}{2(\alpha(N+1)+1)} \tag{2.43}$$

For both cases, we observe that A_T is a decreasing function of N. As an example, the minimum age as a function of N is plotted in Fig. 2.8 for $T = 5$, $T_c = 2$, $\alpha = 1$.

Finally, we note that, when $\alpha \to \infty$, T_c is only used to keep the age $a(t) = 0$,

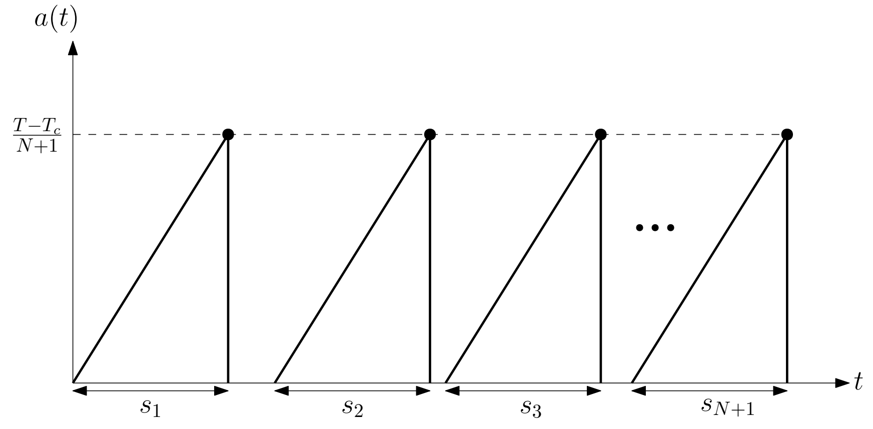


Figure 2.9: Evolution of the optimal age when $\alpha \to \infty$.

and the optimal age can be calculated as:

$$\lim_{\alpha \to \infty} A_T = \frac{1}{2}\left(\frac{T-T_c}{N+1}\right)^2 (N+1) \tag{2.44}$$

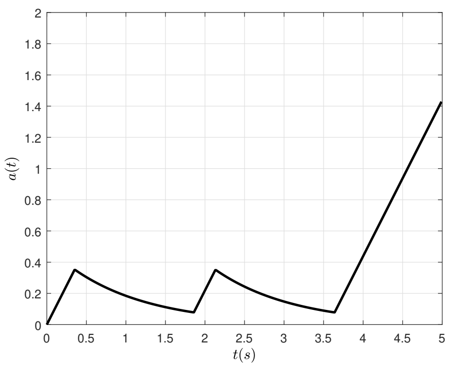
In this case, the optimal age is as shown in Fig. 2.9, which corresponds to the optimal age with instantaneous drops as in the existing literature except for the time intervals where the age stays at zero.[3]

2.5 Numerical Results

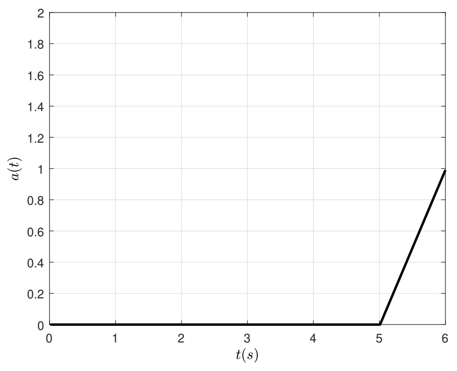
In this section, we give simple numerical examples to illustrate our results. In the first example, we consider the exponentially decaying age model with $T = 5$, $T_c = 3$, $N = 2$ and $\alpha = 1$. Since $T > T_c - \frac{1}{\alpha}$, the optimal update policy is to update $N = 2$ times with equal time allocated to each update, i.e., $c_1 = c_2 = 1.5$. The evolution of $a(t)$ is shown in Fig. 2.10(a).

In the second example, we consider the exponentially decaying age model with

[3]We observe that when $\alpha \to \infty$, the heights of the triangles become the same, which is similar to the result in [62].



(a)



(b)

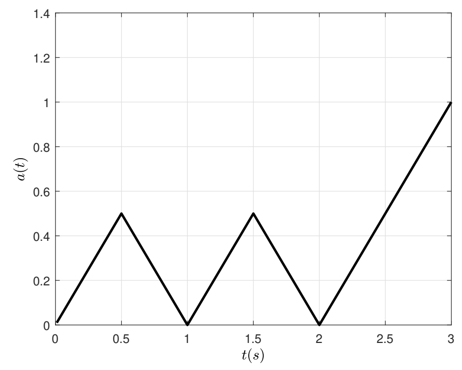
Figure 2.10: Evolution of $a(t)$ in the exponentially decaying age model (a) when $N = 2$, $T = 5$, $T_c = 3$, and $\alpha = 1$, (b) when $N = 2$, $T = 6$, $T_c = 5$, and $\alpha = 1$.

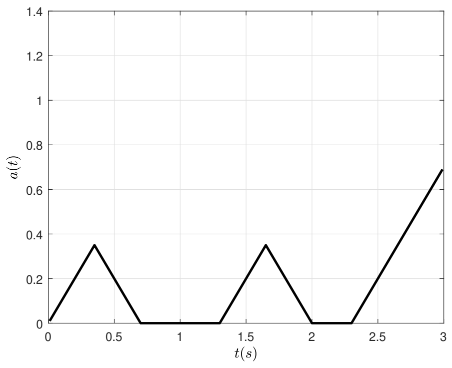
$T = 6$, $T_c = 5$, $N = 2$ and $\alpha = 1$. Since T_c is large enough, i.e., $T \leq T_c - \frac{1}{\alpha}$, the system starts updating at $t = 0$, proceeds to update continuously until T_c, and lets age grow then on until the end. The evolution of $a(t)$ is shown in Fig. 2.10(b).

In the following three examples (third, fourth and fifth), we consider the linearly decaying age model with $\alpha = 1$. In the third example, we see the case where $T_c = \frac{NT}{N(\alpha+1)+\alpha}$. Note that if we have additional updating time, there will be time intervals where the age will stay at zero. The evolution of $a(t)$ is shown in Fig. 2.11(a).

In the fourth example, we consider the case in Lemma 2.5, where $T_c > \frac{NT}{(\alpha+1)(N+1)}$. We see that since T_c is large enough compared to T, some of the total update time is used to make the age zero and for the remaining part of T_c, age will stay at zero which is shown in Fig. 2.11(b).

In the fifth example, we consider the case where $T_c < \frac{NT}{(\alpha+1)(N+1)}$. In this case, age at the end of each update period is equal to zero. Since T_c is small compared to T, in the optimal policy, we do not see any time intervals where the age stays at zero. The evolution of $a(t)$ is shown in Fig. 2.11(c).


(a)


(b)

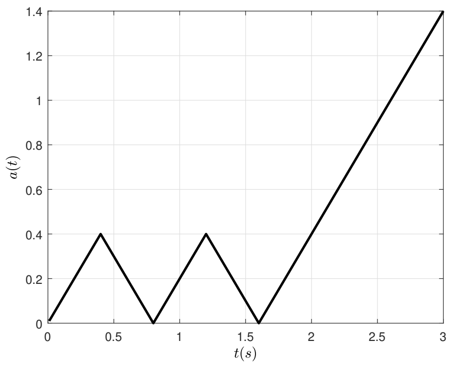
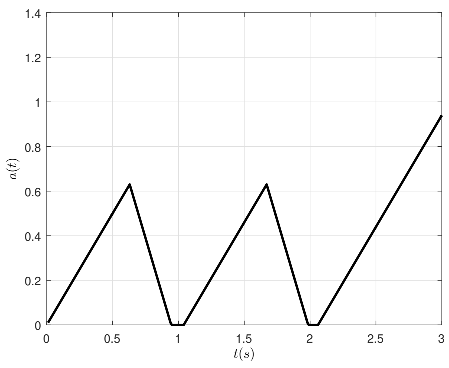

(c)

Figure 2.11: Evolution of $a(t)$ in the linearly decaying age model, for $\alpha = 1$, $N = 2$, $T = 3$, and (a) $T_c = 1$, (b) $T_c = 1.6$, (c) $T_c = 0.8$.


(a)

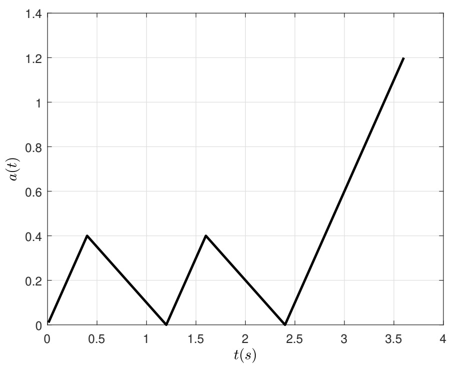

(b)

Figure 2.12: Evolution of $a(t)$ in the linearly decaying age model (a) $\alpha = 2$, $N = 2$, $T = 3$, and $T_c = 0.8$, and (b) $\alpha = 0.5$, $N = 2$, $T = 3.6$, and $T_c = 1.6$.

So far, we have provided examples for the linear case with $\alpha = 1$. In the following examples, we consider the cases with $\alpha > 1$ and $\alpha < 1$. In the first case, we choose $\alpha = 2$, $N = 2$, $T = 3$, and $T_c = 0.8$, and in the second case, we choose $\alpha = 0.5$, $N = 2$, $T = 3.6$, and $T_c = 1.6$. The optimal policies are shown in Fig. 2.12(a) and Fig. 2.12(b), respectively.

2.6 Conclusion

In this chapter, we introduced the concept of soft updates which is relevant in systems with human interactions and social media settings, where the decrease in age happens gradually over soft update periods. We study two soft update regimes: in the first one, age decays exponentially and in the second one age decays linearly during the soft update period. In both models, we showed that the optimal policy is to have N updates and T_c should be completely utilized with allocating equal amount of time for each update.

CHAPTER 3

Age of Information with Distortion

3.1 Introduction

In this chapter, we consider an information update system where an information receiver requests updates from an information provider in order to minimize the age of information at the receiver. To generate an update, the information provider completes a set of tasks such as collecting data and processing them. We consider the *quality* of updates via their *distortion*. We model the *quality* (resp., the *distortion*) of an update as a monotonically increasing (resp., monotonically decreasing) function of the processing time spent to generate the update at the transmitter.

Examples of such systems can be found in sensor networking and distributed computation applications. For instance, in a sensor networking application where multiple sensors observe the realization of a common underlying random variable (e.g., temperature), if the information provider generates an update using the observation of a single sensor, the update will be generated faster, but will have large distortion; and conversely, if the information provider generates an update using the observations of all sensors, the update will be generated with a delay, but will have

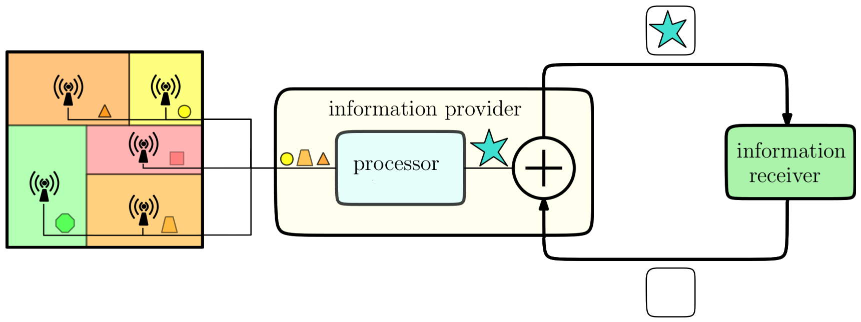


Figure 3.1: An information updating system which consists of an information provider which collects/processes data and an information receiver.

small distortion. Similarly, in a distributed computation system with stragglers, the master can generate an update using faster servers with lower quality, or utilize all servers to generate a better quality update with a delay. Thus, there is a trade-off between processing time and quality.

We consider the information update system shown in Fig. 3.1. The information provider connects to multiple units (sensors, servers, etc.) to generate an update. When there is no update, the information at the receiver gets stale over time, i.e., the age increases linearly. The information receiver requests an update from the information provider. After receiving the update request, the information provider allocates c_i amount of time as shown in Fig. 3.2 for processing the information. During this processing time, the information used to generate the update ages by c_i. When the information provider sends the update to the receiver, the age at the receiver decreases down to the age of the update which is c_i, as the communication time between the transmitter and the receiver is negligible.

We model distortion as a monotonically decreasing function of processing time, c_i, motivated by the diminishing returns property [140]. We consider exponentially

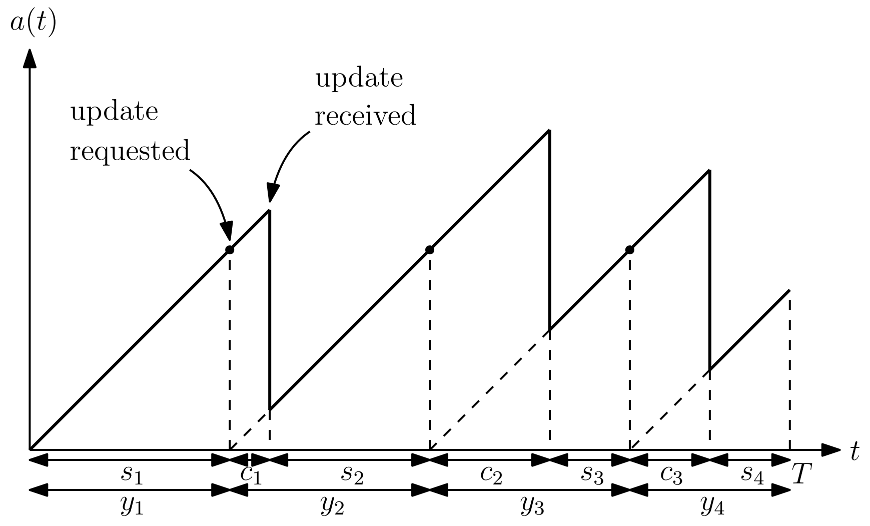


Figure 3.2: Age evolution at the receiver.

and inverse linearly decaying distortion functions as examples. In particular, inverse linearly decaying distortion function arises in sensor networking applications, where all sensors observe an underlying random variable distorted by independent Gaussian noise, and the information provider combines sensor observations linearly to minimize the mean squared error (see Section 3.2).

In this chapter, we determine age-optimum updating schemes for a system with a distortion constraint on each update. We are given a total time duration over which the average age is calculated T, the total number of updates N, the maximum allowed distortion as a function of the current age $f(y)$, and the distortion function as a function of the processing time $D(c)$. We solve for the optimum request times for the updates at the receiver and the optimum processing times of the updates at the transmitter, to minimize the overall age.

In this work, we consider the general case where the distortion constraint is a function of the processing time at the transmitter and the current age at the receiver.[1] Distortion function is always monotonically decreasing with the processing time. Regarding the dependence of the distortion constraint on the current age at the receiver, we consider three different scenarios: First, as in [108], distortion constraint is constant (independent of the current age), second, the distortion constraint is inversely proportional with the current age, and third, the distortion constraint is proportional with the current age. The second case is motivated by the following observation: If the age at the receiver is high, the receiver may want to receive a high quality update, i.e., an update with low distortion, to replace its current information with more accurate information. In this case a high age implies a low desired distortion, hence, age and distortion constraints are inversely proportional. The third case is motivated by the following observation: If the age at the receiver is high, the receiver may want to receive a quick update, i.e., an update with high distortion, to replace its current information with a fresh information. In this case, the receiver trades its obsolete but high quality update with a fresh but low quality update. This may be desirable in applications where the freshness of information matters more than the quality of the information. Therefore, in this work, we consider the cases where the distortion constraint is 1) a constant, 2) a decreasing, and 3) an increasing function of the current age.

In this chapter, we provide the age-optimal policies by finding the optimum processing times and the optimum update request times. We show that the optimum processing time is always equal to the minimum required processing time that meets

[1]In the conference version of this work in [108], we considered the simpler case where the distortion constraint was a function of the processing time only, i.e., it was not a function of the current age.

the distortion constraint. If there is no active constraint on distortion, i.e., the distortion constraint is high enough, the optimum processing time is equal to zero. We observe three different optimum policies for update request times depending on the level of distortion constraint. When the distortion constraint is large enough except in the case where the distortion function is inversely proportional to the current age, we show that the optimal policy is to request updates with equal inter-update times. When the distortion constraint is relatively large, i.e., the required processing time is relatively small compared to the total time period, it is optimal to request updates regularly following a waiting (request) time after receiving each update, with a longer request time for the first update than others. When the distortion constraint is relatively small, i.e., the required processing time is relatively large compared to the total time period, the optimal policy is to request an update once the previous update is received, i.e., back-to-back, except for a potentially non-zero requesting time for the first update.

3.1.1 Related Work

References that are most closely related to our work are [68, 69, 72, 84, 87, 91, 96, 141, 142], which consider the trade-off between service performance and information freshness. [91] emphasizes the difference between service completion time and the age. [68] considers the joint optimization of information freshness, quality of information, and total energy consumption which assumes that the distortion (utility) function follows law of diminishing returns and models the age and energy cost as convex functions. The main contribution of [68] is deriving an online algorithm which is 2-competitive. In our work, there is no explicit energy constraint, but the total number of updates N for a given total time duration T is limited. Even though we consider the age and quality of the updates, the problem settings are different where we minimize the average age of information, which is inherently non-convex, subject to a distortion constraint for each update. Furthermore, we consider age-dependent distortion constraint which also differentiates our overall work from [68].

In [69], service performance is measured by how *quickly* the provider responds to the queries of the receiver. In [69], the performance of the system is considered to be the highest when the service provider responds immediately upon a request. In [69], by responding quickly, the service provider may be using available, but perhaps outdated, information resulting in larger age; on the other hand, if the provider waits for processing new data and responds to the queries a bit later, information of the update may be fresher. Thus, in the model of [69], processing data degrades quality of service as it worsens response time, but improves the age. In contrast, in our model, processing data improves service performance (the quality of updates), but worsens the age, as the age at the receiver grows while the transmitter processes the data. Thus, the models and trade-offs captured in [69] and here are substantially different.

As we model the distortion as a function of the processing time and the maximum allowed distortion as a function of the instantaneous age, update duration depends on the current age. A similar problem with age-dependent update duration was considered in [96] where the solution for a relaxed and simplified version of the original problem was given. Different from [96], where only the case in which the update duration is proportional to the current age is considered, here we consider the cases in which the update duration is proportional and inversely proportional with the current age, and we provide exact solutions for both problems.

References [72, 141] show that scheduling based on value of information (VoI) improves the service performance. In [72], the VoI measures the amount of uncertainty reduction in the process at the information receiver. [72] expresses VoI as a function of AoI, and designs a scheduler for AoI and another one for VoI. By comparing the performances of these two schedulers, [72] shows that scheduling based on VoI results in lower uncertainty, and therefore higher control performance compared to scheduling based on AoI. [141] proposes an index policy to calculate the VoI of the update packets where the VoI of a packet decreases with the age and increases with the precision of the source, and shows that the optimal policy which minimizes the estimation error is to schedule the update with the largest VoI.

The problem which considers the trade-off between video freshness and the video quality in real time systems in [142] is a specific application of our model.[2] In [142], the original video is encoded into multiple layers. With the first layer, a low quality video can be decoded. With the remaining layers, the video quality at the users can be enhanced. If the videos are updated infrequently, then the users

[2]Similar to [143], we can consider the following quantization problem that is applicable to our work. Assuming that the transmitter can get one bit from the sensor at a time, a quantized status update with c_i bits at the sensor can be sent to transmitter after c_i amount of time (denoted as processing time). Thus, if we increase quantization levels (which also increases the processing time), the transmitter can get higher quality updates, but the received updates can be obsolete. On the other hand, the transmitter can get a quick update with smaller number of quantization levels, but the received updates will have higher distortion.

can collect more layers. Thus, the video quality at the users will be high, but the received videos can be obsolete. On the other hand, if the transmitter updates videos frequently, the users may receive fresh but lower quality videos as the users may not collect all the layers. The aim of [142] is to maximize the overall average utility which is a combination of freshness and the quality of the videos at the users within a total time duration. We note that the quality function in [142] can be equivalently represented by the distortion function in our work. Different from [142], our aim is to minimize average AoI subject to the distortion constraints on each update which can be age dependent.

Finally, [84] and [87] consider *partial updates* where the information content is smaller compared to full updates, which also resembles trading-off update quality with service time.

3.2 System Model and Problem Formulation

Let $a(t)$ be the instantaneous age at time t, with $a(0) = 0$. When there is no update, the age increases linearly over time; see Fig. 3.2. When an update is received, the age at the receiver decreases down to the age of the latest received update. The channel between the information provider and the receiver is assumed to be perfect with zero transmission times, as in e.g., [37, 38, 43, 44]. However, in order to generate an update, the provider needs to allocate a processing time. For update i, the provider allocates c_i amount of processing time.

We model the distortion function as a monotonically decreasing function of processing time due to the diminishing returns property. For instance, we consider an exponentially decaying distortion function, D_e,

$$D_e(c_i) = a\left(e^{-bc_i} - d\right), \tag{3.1}$$

where $d \leq e^{-bc_{max}}$ so that the distortion function is always nonnegative. In addition, we consider an inverse linearly decaying distortion function, D_ℓ,

$$D_\ell(c_i) = \frac{a}{bc_i + d}, \tag{3.2}$$

which arises in sensor networking applications.[3] In particular, consider a system with M sensors placed in an area, measuring a common random variable X with mean μ_X and variance σ_X^2. The measurement at each sensor, Y_j, is perturbed by an i.i.d. zero-mean Gaussian noise with variance σ^2. Information provider uses a linear estimator, $\hat{X} = \sum_{j=1}^{M} w_j Y_j$ to minimize the distortion (mean squared error) defined as $D_\ell = \mathbb{E}[(\hat{X} - X)^2]$. In this model, we assume that the information provider connects to one sensor at a time and spends one unit of time to retrieve the measurement from that sensor. Thus, if the information provider connects to c_i sensors, it spends c_i units of time for processing (i.e., retrieving data) and achieves a distortion of $D_\ell(c_i) = \sigma^2/(c_i + \frac{\sigma^2}{\mu_X^2 + \sigma_X^2})$ for the ith update, which has the inverse linearly decaying form in (3.2).

[3]Other forms of distortion may be considered as well. For example, for a distributed computation system, one can consider a model with a non-zero cold start computation time, during which the distortion is infinite, and once computations are received from all servers, the distortion becomes zero.

Let s_i be the time interval between the reception time of the $(i-1)$th update and the request time of the ith update at the receiver, and let c_i be the processing time of the ith update at the transmitter; see Fig. 3.2. Then, $y_i = s_i + c_{i-1}$ is the time interval between requesting the $(i-1)$th and the ith updates; it is also the age at the time of requesting the ith update; see Fig. 3.2. The remaining time after receiving the last update is s_{N+1}, i.e., $s_{N+1} = T - \sum_{i=1}^{N}(s_i + c_i)$, and $c_0 = 0$.

We define $f(y_i)$ as the maximum allowed distortion for each update where y_i is the current age. We will start with the case where the maximum allowed distortion is a constant, $f(y_i) = \beta$, i.e., it does not depend on the current age, and then continue with the general case where it explicitly depends on the current age. We consider two sub-cases in the latter case. In the first sub-case, the maximum allowed distortion decreases with the current age, and in the second sub-case, the maximum allowed distortion increases with the current age.

Our objective is to minimize the average age of information at the information receiver over a total time period T, subject to having a desired level of distortion for each update, given that there are N updates. We formulate the problem as,

$$\begin{aligned}
\min_{\{s_i, c_i\}} \quad & \frac{1}{T}\int_0^T a(t)dt \\
\text{s.t.} \quad & \sum_{i=1}^{N+1} s_i + c_{i-1} = T \\
& D(c_i) \leq f(y_i), \quad i = 1, \dots, N \\
& s_i \geq 0, \quad c_i \geq 0,
\end{aligned} \tag{3.3}$$

where $a(t)$ is the instantaneous age, $D(c_i)$ is the distortion function which is monotonically decreasing in c_i, and $f(y_i)$ is the maximum allowed distortion function for update i as a function of the current age y_i. We solve the optimization problem in (3.3) by determining the optimum update request times after the previous update is delivered, s_i, and the optimum update processing times, c_i. The distortion function $D(c_i)$ may be $D_e(c_i)$ or $D_\ell(c_i)$ defined above, or any other appropriate distortion function depending on the application. The maximum allowed distortion $f(y_i)$ may be constant, i.e., $f(y_i) = \beta$, or it may be a function of the current age y_i. We consider two specific cases where $f(y_i)$ is a decreasing function of y_i and where $f(y_i)$ is an increasing function of y_i. Let $A_T \triangleq \int_0^T a(t)dt$ be the total age. Note that minimizing $\frac{A_T}{T}$ is equivalent to minimizing A_T since T is a known constant.

With these definitions, and using the age evolution curve in Fig. 3.2, the total age A_T is,

$$A_T = \frac{1}{2}\sum_{i=1}^{N+1}(s_i + c_{i-1})^2 + \sum_{i=1}^{N} c_i(s_i + c_{i-1}). \tag{3.4}$$

In the following section, we provide the optimal solution for the problem defined in (3.3) when the maximum allowed distortion is constant.

3.3 Constant Allowable Distortion

In this section, we consider the case $f(y_i) = \beta$. Since $D(c_i)$ is a monotonically decreasing function of c_i, $D(c_i) \leq \beta$ is equivalent to $c_i \geq c$ where $c = D^{-1}(\beta)$ is a constant. Thus, we replace the distortion constraint given in (3.3) with $c_i \geq c$. In addition, we substitute $y_i = s_i + c_{i-1}$ for $i = 1, \ldots, N+1$. Then, using (3.4), we rewrite the problem in (3.3) as,

$$\begin{aligned} \min_{\{y_i, c_i\}} \quad & \frac{1}{2}\sum_{i=1}^{N+1} y_i^2 + \sum_{i=1}^{N} c_i y_i \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_1 \geq 0, \quad y_i \geq c_{i-1}, \quad i = 2, \ldots, N+1 \\ & c_i \geq c, \quad i = 1, \ldots, N. \end{aligned} \tag{3.5}$$

The optimization problem in (3.5) is not convex due to the multiplicative terms involving c_i and y_i. We note that $c_i = c$ for $i = 1, \ldots, N$ is an optimum selection, since this selection minimizes the second term in the objective function and at the same time yields the largest feasible set for the remaining set of variables (i.e., y_is) in the problem in (3.5). Thus, the optimization problem in (3.5) becomes,

$$\begin{aligned} \min_{\{y_i\}} \quad & \frac{1}{2}\sum_{i=1}^{N+1} y_i^2 + \sum_{i=1}^{N} c y_i \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_1 \geq 0, \quad y_i \geq c, \quad i = 2, \ldots, N+1, \end{aligned} \tag{3.6}$$

which is now only in terms of y_i.

When $\beta = \infty$, and thus, $c = 0$ in (3.6), i.e., there is no active distortion constraint, the optimal solution is to choose $y_i = \frac{T}{N+1}$ for all i. Therefore, for the rest of this section, we consider the case where $\beta < \infty$, and thus, $c > 0$.

We write the Lagrangian for the problem in (3.6) as,

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{N+1} y_i^2 + \sum_{i=1}^{N} cy_i - \lambda\left(\sum_{i=1}^{N+1} y_i - T\right) - \sum_{i=2}^{N+1}\theta_i(y_i - c) - \theta_1 y_1, \qquad (3.7)$$

where $\theta_i \geq 0$ and λ can be anything. The problem in (3.6) is convex. Thus, the KKT conditions are necessary and sufficient for the optimal solution. The KKT conditions are,

$$\frac{\partial \mathcal{L}}{\partial y_i} = y_i + c - \lambda - \theta_i = 0, \quad i = 1, \dots, N, \qquad (3.8)$$

$$\frac{\partial \mathcal{L}}{\partial y_{N+1}} = y_{N+1} - \lambda - \theta_{N+1} = 0. \qquad (3.9)$$

The complementary slackness conditions are,

$$\lambda\left(\sum_{i=1}^{N+1} y_i - T\right) = 0, \qquad (3.10)$$

$$\theta_1 y_1 = 0, \qquad (3.11)$$

$$\theta_i(y_i - c) = 0, \quad i = 2, \dots, N+1. \qquad (3.12)$$

When $y_i > c$ for all i, we have $\theta_i = 0$ due to (3.11) and (3.12). Then, from (3.8) and (3.9), we obtain $y_i = \lambda - c$ for $i = 1, \dots, N$, and $y_{N+1} = \lambda$. Since $\sum_{i=1}^{N+1} y_i = T$, we find $\lambda = \frac{T+Nc}{N+1}$. Thus, the optimal solution becomes,

$$y_i = \frac{T - c}{N+1}, \quad i = 1, \dots, N, \qquad (3.13)$$

$$y_{N+1} = \frac{T + Nc}{N+1}. \qquad (3.14)$$

In order to have $y_i > c$, we need $T > (N+2)c$. Viewing this condition from the perspective of c, this is the case when c is small in comparison to T. Therefore, we note that, in this case, when minimum processing time, c, is relatively small, the optimal policy is to choose y_i as equal as possible except for y_{N+1}. When c becomes larger compared to T, $y_i - c$ decreases. Specifically, when $T = (N+2)c$, $y_i = c$ for $i = 1, \ldots, N$.

In the remaining case, i.e., when $T < (N+2)c$, $y_1 < c$ and $y_{N+1} > c$, we have $\theta_1 = 0$ and $\theta_{N+1} = 0$ by (3.11) and (3.12). Then, by solving $y_i = \lambda - c$, $y_{N+1} = \lambda$, and $\sum_{i=1}^{N+1} y_i = T$, we obtain,

$$y_1 = \frac{T - Nc}{2}, \tag{3.15}$$

$$y_i = c, \quad i = 2, \ldots, N, \tag{3.16}$$

$$y_{N+1} = \frac{T - (N-2)c}{2}. \tag{3.17}$$

Since $y_1 > 0$, we need $Nc < T$. Thus, this solution applies when $Nc < T \leq (N+2)c$.

Finally, when $T = Nc$, the optimal solution becomes,

$$y_1 = 0, \tag{3.18}$$

$$y_i = c, \quad i = 2, \ldots, N+1. \tag{3.19}$$

In summary, when $c = 0$, i.e., we do not have any distortion constraints, then the optimal solution is to update in every $\frac{T}{N+1}$ units of time, i.e., $y_i = \frac{T}{N+1}$ for all i. When $c > 0$ but, relatively small compared to T, i.e., $(N+2)c < T$, the optimal solution is to wait for $\frac{T-c}{N+1}$ to request the first update. For the remaining updates, the receiver waits for $\frac{T-(N+2)c}{N+1}$ time to request another update after the previous update is received. After requesting N updates, the optimal policy is to let the age grow for the remaining $\frac{T-c}{N+1}$ units of time. When c becomes large compared to T, i.e., $Nc < T \leq (N+2)c$, the optimal policy is to wait for $\frac{T-Nc}{2}$ to request the first update and request the remaining updates as soon as the previous update is received, i.e., back-to-back. After updating N times, we let the age grow for the remaining $\frac{T-Nc}{2}$ units of time. Finally, when $T = Nc$, the optimal policy is to request the first update at $t = 0$ and request the remaining updates as soon as the previous update is received, i.e., back-to-back. We note that when $Nc > T$, there is no feasible policy. The possible optimal policies are shown in Fig. 3.3.

In the following section, we provide the optimal solution for the problem defined in (3.3) when the maximum allowed distortion is age-dependent.

3.4 Age-Dependent Allowable Distortion

In this section, we consider the case where the maximum allowed distortion $f(y_i)$ depends explicitly on the instantaneous age y_i. As motivated in the introduction section, this dependence may take different forms. In particular, depending on the application, $f(y_i)$ may be a decreasing or an increasing function of y_i. In the following two sub-sections, we consider two sub-cases: when $f(y_i)$ is inversely proportional to y_i and when $f(y_i)$ is proportional to y_i.

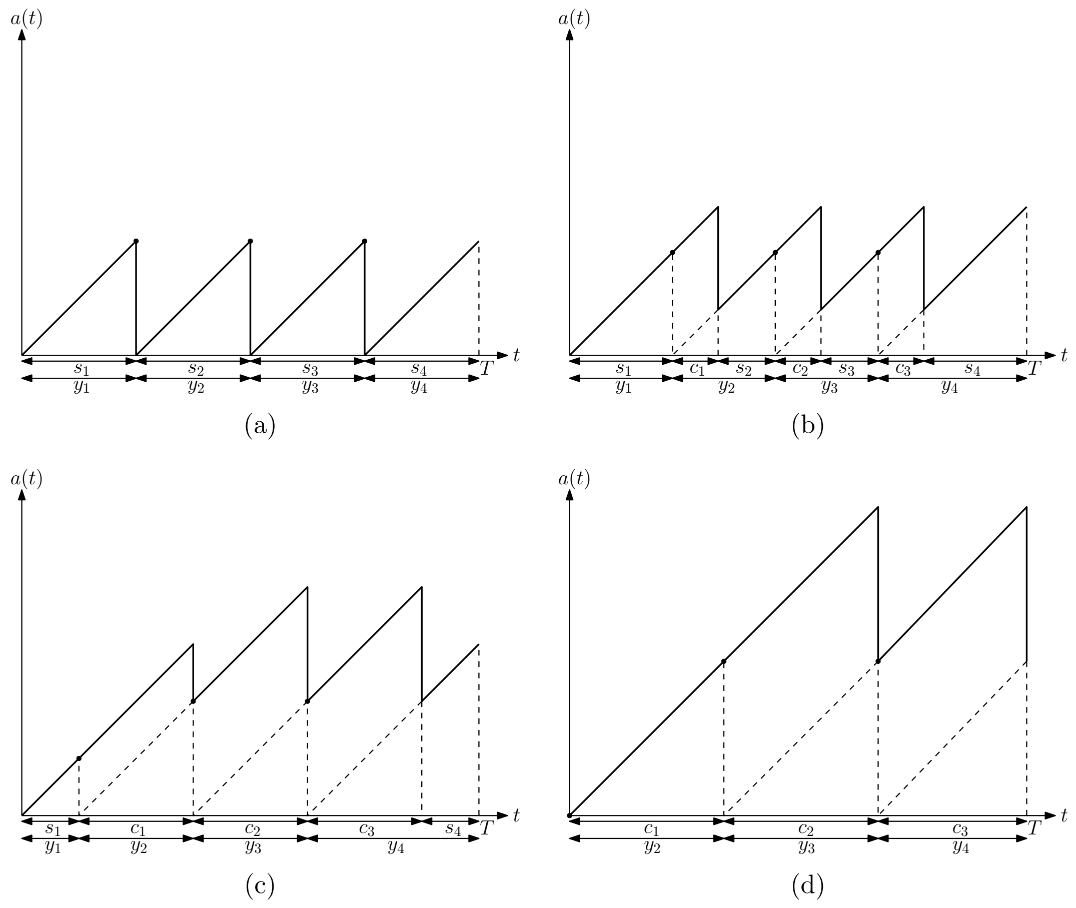


Figure 3.3: Evolution of $a(t)$ with optimal update policies when the distortion function does not depend on the current age in the case of (a) $c = 0$, (b) $c > 0$ and $(N+2)c < T$, (c) $Nc < T \leq (N+2)c$, (d) $T = Nc$.

3.4.1 Allowable Distortion is Inversely Proportional to the Instantaneous Age

We consider the case where $f(y_i)$ is a decreasing function of y_i. Since the distortion function $D(c_i)$ is a decreasing function of the processing time c_i, the distortion constraint for each update, i.e., $D(c_i) \leq f(y_i)$, becomes $c_i \geq D^{-1}(f(y_i))$ where $D^{-1}(\cdot)$ is the inverse function of the distortion function. As $f(y_i)$ is a decreasing function of y_i, the minimum required processing time $D^{-1}(f(y_i))$ is an increasing function of the current age y_i, i.e., we have $D^{-1}(f(y_j)) \geq D^{-1}(f(y_i))$ for all $y_j \geq y_i$. In general, $D^{-1}(f(y_i))$ function can be arbitrary depending on the selections of $D(c_i)$ and $f(y_i)$. However, in order to make the analysis tractable, in this chapter, we focus on a particular case where the distortion constraint for each update in (3.3), i.e., $D(c_i) \leq f(y_i)$, implies $c_i \geq \alpha y_i$, where α is a positive constant. An example for this case is obtained, if we consider the inverse linearly decaying distortion function, $D_\ell(c_i) = \frac{a}{bc_i+d}$ in (3.2), and use an inverse linearly decaying allowable distortion function $f(y_i) = \frac{a}{\kappa y_i+d}$.

The optimization problem in (3.3) in this case becomes,

$$\begin{aligned} \min_{\{y_i,c_i\}} \quad & \frac{1}{2}\sum_{i=1}^{N+1} y_i^2 + \sum_{i=1}^{N} c_i y_i \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_1 \geq 0, \quad y_i \geq c_{i-1}, \quad i = 2, \dots, N+1 \\ & c_i \geq 0, \quad c_i \geq \alpha y_i, \quad i = 1, \dots, N. \end{aligned} \tag{3.20}$$

In the following lemma, we show that the processing time for each update should be equal to the minimum required time in order to satisfy the distortion constraint, i.e., $c_i = \alpha y_i$, for all i.

Lemma 3.1 *In the age-optimal policy, processing time for each update is equal to the minimum required time which meets the distortion constraint with equality, i.e., $c_i = \alpha y_i$ for all i.*

Proof: Let us assume that on the contrary there exists an optimal policy such that $c_j > \alpha y_j$ for some j. Then, we find another feasible policy denoted by $\{s'_i, c'_i\}$ such that $c'_j = c_j - \epsilon$, $s'_{j+1} = s_{j+1} + \epsilon$ and $y'_{j+1} = s'_{j+1} + c'_j = y_{j+1}$. Since $c_j > \alpha y_j$, we can always choose sufficiently small ϵ so that we have $c'_j \geq \alpha y'_j$ for the new policy. We have $y_i = y'_i$ for all i and $c_i = c'_i$ for $i \neq j$ which means that in the new policy, we keep all other variables the same except for c'_j and s'_{j+1}. Inspecting the objective function of (3.20), we note that in the new policy, the age is decreased by ϵy_j. Since the new policy with $\{s'_i, c'_i\}$ achieves a smaller age, we reach a contradiction. Therefore, in the age-optimal policy, we must have $c_i = \alpha y_i$, for all i. ■

Intuitively, as the age of the receiver and the generated update increase during an update generation process, age-optimal policy is achieved when the processing time is equal to the minimum required processing time. We remark that Lemma 3.1 provides an alternative proof for the fact that c_i must be such that $c_i = c$ in (3.5). We argued this briefly after (3.5) based on the observation that this selection minimizes the objective function and enlarges the feasible set.

Using Lemma 3.1, we let $c_i = \alpha y_i$, and rewrite (3.20) as,

$$\begin{aligned} \min_{\{y_i\}} \quad & \left(\frac{1}{2} + \alpha\right) \sum_{i=1}^{N} y_i^2 + \frac{1}{2} y_{N+1}^2 \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_1 \geq 0, \quad y_i \geq \alpha y_{i-1}, \quad i = 2, \dots, N+1, \end{aligned} \tag{3.21}$$

which is only in terms of y_i.

We write the Lagrangian for the problem in (3.21) as,

$$\mathcal{L} = \left(\frac{1}{2}+\alpha\right)\sum_{i=1}^{N} y_i^2 + \frac{1}{2}y_{N+1}^2 - \lambda\left(\sum_{i=1}^{N+1} y_i - T\right) - \beta_1 y_1 - \sum_{i=2}^{N+1}\beta_i(y_i - \alpha y_{i-1}), \tag{3.22}$$

where $\beta_i \geq 0$ and λ can be anything. The problem in (3.21) is convex. Thus, the KKT conditions are necessary and sufficient for the optimal solution. The KKT conditions are,

$$\frac{\partial \mathcal{L}}{\partial y_i} = (1+2\alpha)y_i - \lambda - \beta_i + \alpha\beta_{i+1} = 0, \quad i = 1,\dots,N, \tag{3.23}$$

$$\frac{\partial \mathcal{L}}{\partial y_{N+1}} = y_{N+1} - \lambda - \beta_{N+1} = 0. \tag{3.24}$$

The complementary slackness conditions are,

$$\lambda\left(\sum_{i=1}^{N+1} y_i - T\right) = 0, \tag{3.25}$$

$$\beta_1 y_1 = 0, \tag{3.26}$$

$$\beta_i(y_i - \alpha y_{i-1}) = 0, \quad i = 2,\dots,N+1. \tag{3.27}$$

First, we consider the case where $s_i > 0$ and $c_i > 0$ for all i. Then, we have $y_1 > 0$ and $y_i > \alpha y_{i-1}$ for all $i = 2,\dots,N+1$. The former statement follows because $y_1 = s_1 > 0$, and the latter statement follows because $y_i = \alpha c_i$ due to Lemma 3.1 and $y_i = s_i + c_{i-1} = s_i + \alpha y_{i-1} > \alpha y_{i-1}$ since $s_i > 0$. Thus, from (3.26)-(3.27), we have $\beta_i = 0$ for all i. By using (3.23)-(3.24), we have $y_i = \frac{\lambda}{2\alpha+1}$ for $i = 1,\dots,N$, and

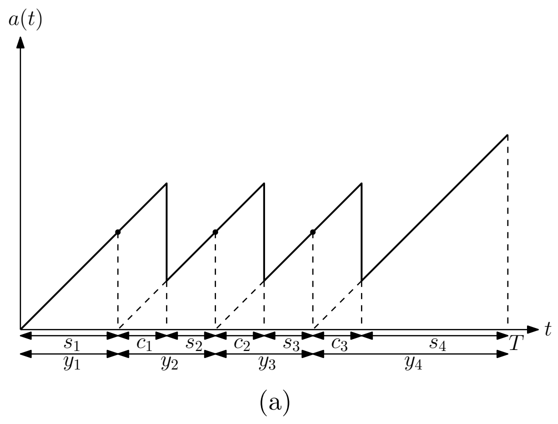


Figure 3.4: Age evolution at the receiver when $f(y_i)$ is inversely proportional to the current age for (a) $\alpha \leq 1$ and (b) $\alpha > 1$.

$y_{N+1} = \lambda$. Since $\sum_{i=1}^{N+1} y_i = T$ from (3.25), we find $\lambda = \frac{(2\alpha+1)T}{N+2\alpha+1}$. Thus, the optimal solution in this case is,

$$y_i = \frac{T}{N+2\alpha+1}, \quad i = 1, \ldots, N, \tag{3.28}$$

$$y_{N+1} = \frac{(2\alpha+1)T}{N+2\alpha+1}. \tag{3.29}$$

In order to satisfy $y_i > \alpha y_{i-1}$, we need $\alpha < 1$. A typical age evolution curve for $\alpha < 1$ is shown in Fig. 3.4(a). When $\alpha = 1$, we note that the optimal solution follows (3.28) and (3.29), but $y_i = \alpha y_{i-1}$ for $i = 2, \ldots, N$.

Next, we find the optimal solution for $\alpha > 1$. If we have only the total time constraint, then the optimal solution is to choose y_is equal for $i = 1, \ldots, N$. Since $\alpha > 1$, we cannot choose y_is equal due to $y_i \geq \alpha y_{i-1}$ constraints. In the following lemma, we prove that when $\alpha > 1$, $y_i = \alpha y_{i-1}$ for $i = 2, \ldots, N$.

Lemma 3.2 *When $\alpha > 1$, we have $y_i = \alpha y_{i-1}$ for $i = 2, \ldots, N$.*

Proof: Assume for contradiction that there exists an age-optimal policy with $y_j > \alpha y_{j-1}$ for some $j \in \{2, \dots, N\}$. From (3.27), we have $\beta_j = 0$. From (3.23), we get $y_j = \frac{\lambda - \alpha B_{j+1}}{2\alpha+1}$ and $y_{j-1} = \frac{\lambda+\beta_{j-1}}{2\alpha+1}$. Since $y_j \geq 0$, we must have $\lambda \geq 0$. By using $y_j > \alpha y_{j-1}$, we must have $(1-\alpha)\lambda > \alpha(\beta_{j+1} + \beta_{j-1})$. Since $\alpha > 1$ and $\lambda \geq 0$, this implies $(1-\alpha)\lambda \leq 0$, which further implies $\alpha(\beta_{j+1} + \beta_{j-1}) < 0$. However, this inequality cannot be satisfied since $\beta_i \geq 0$ for all i. Thus, we reach a contradiction and in the age-optimal policy, we must have $y_i = \alpha y_{i-1}$ for $i = 2, \dots, N$. ■

Then, the optimal policy is in the form of $y_i = \alpha^{i-1}\eta$ for $i = 1, \dots, N$ and $y_{N+1} = T - \sum_{i=1}^{N} y_i$ where η is a constant. We write the total age in terms of η as,

$$A_T(\eta) = \left(\frac{1}{2} + \alpha\right)\eta^2 \left(\frac{\alpha^{2N} - 1}{\alpha^2 - 1}\right) + \frac{1}{2}\left(T - \left(\frac{\alpha^N - 1}{\alpha - 1}\right)\eta\right)^2. \tag{3.30}$$

In order to find the optimal η, we differentiate (3.30), which is quadratic in η, with respect to η and equate it to zero. We find the optimal solution for $\alpha > 1$ as,

$$y_1 = \frac{T(\alpha^{N+2} - \alpha^N - \alpha^2 + 1)}{2(\alpha^{2N+2} - \alpha^{N+1} - \alpha^N - \alpha^2 + \alpha + 1)}, \tag{3.31}$$

$$y_i = \alpha^{i-1} y_{i-1}, \quad i = 2, \dots, N, \tag{3.32}$$

$$y_{N+1} = T - \sum_{i=1}^{N} y_i. \tag{3.33}$$

A typical age evolution curve for $\alpha > 1$ is shown in Fig. 3.4(b).

In summary, when $\alpha < 1$, i.e., when the required processing time increases slowly with the age, then the optimal policy is to request the updates regularly following a waiting time after receiving each update as shown in Fig. 3.4(a). When

$\alpha > 1$, i.e., when the required processing time increases rapidly with the age, then the optimal policy is to request the updates once the previous update is delivered (except for a positive waiting time for the first update) as shown in Fig. 3.4(b).

3.4.2 Allowable Distortion is Proportional to the Instantaneous Age

We consider the case where $f(y_i)$ is an increasing function of y_i. Similar to Section 3.4.1, the distortion constraint for each update, i.e., $D(c_i) \leq f(y_i)$, is equivalent to $c_i \geq D^{-1}(f(y_i))$. As $f(y_i)$ is an increasing function of y_i, the minimum required processing time $D^{-1}(f(y_i))$ is a decreasing function of the current age y_i, i.e., we have $D^{-1}(f(y_j)) \leq D^{-1}(f(y_i))$ for all $y_j \geq y_i$. Even though $D^{-1}(f(y_j))$ can be arbitrary, in this chapter, in order to make the analysis tractable, we focus on a specific case where the distortion constraint for each update in (3.3), i.e., $D(c_i) \leq f(y_i)$, implies $c_i \geq c - \alpha y_i$. In this section, we assume $\alpha < \frac{1}{2}$. An example of this case is obtained, if we consider the inverse linearly decaying distortion, $D_\ell(c_i) = \frac{a}{bc_i+d}$ in (3.2), and use $f(y_i) = \frac{a}{u-\kappa y_i}$. Thus, while the distortion constraint in Section 3.4.1 was $c_i \geq \alpha y_i$, the distortion constraint in this section is $c_i \geq c - \alpha y_i$.

The optimization problem in (3.3) in this case becomes,

$$\begin{aligned} \min_{\{y_i, c_i\}} \quad & \frac{1}{2}\sum_{i=1}^{N+1} y_i^2 + \sum_{i=1}^{N} c_i y_i \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_1 \geq 0, \quad y_i \geq c_{i-1}, \quad i = 2, \dots, N+1 \\ & c_i \geq 0, \quad c_i \geq c - \alpha y_i, \quad i = 1, \dots, N. \end{aligned} \tag{3.34}$$

In the following lemma, we show that the processing time for each update should be equal to the minimum processing time which satisfies the distortion constraint, i.e., $c_i = (c - \alpha y_i)^+$ for $i = 1, \dots, N$, where $(x)^+ = \max\{0, x\}$.

Lemma 3.3 *In the age-optimal policy, processing time for each update is equal to the minimum required time which meets the distortion constraint with equality, i.e.,* $c_i = (c - \alpha y_i)^+$*, for all* i*.*

Proof: Let us assume for contradiction that there exists an optimal policy such that $c_j > c - \alpha y_j$ for some j. If $y_j < \frac{c}{\alpha}$, then we find another feasible policy denoted by $\{s_i', c_i'\}$ such that $c_j' = c_j - \epsilon$, $s_{j+1}' = s_{j+1} + \epsilon$ and $y_{j+1}' = s_{j+1}' + c_j' = y_{j+1}$. Since $c_j > c - \alpha y_j$, we can always choose sufficiently small ϵ so that we have $c_j' \geq c - \alpha y_j'$ for the new policy. We have $y_i = y_i'$ for all i and $c_i = c_i'$ for $i \neq j$ which means that in the new policy, we keep all other variables the same except c_j' and s_{j+1}'. We note that in the new policy, age is decreased by ϵy_j. Since the new policy with $\{s_i', c_i'\}$ achieves a smaller age, we reach a contradiction. Therefore, in the age-optimal policy, we must have $c_i = c - \alpha y_i$ for all i when $y_i < \frac{c}{\alpha}$. If $y_j \geq \frac{c}{\alpha}$, then $c_j \geq 0$ is the only constraint on c_j. If $c_j > 0$, we can similarly argue that decreasing c_j further reduces the age until c_j becomes zero. Thus, we reach a contradiction and when $y_j \geq \frac{c}{\alpha}$, in the optimal solution, we must have $c_j = 0$. By combining these two parts, we conclude that in the optimal policy, we must have $c_i = (c - \alpha y_i)^+$, for $i = 1, \dots, N$.

■

Using Lemma 3.3, we let $c_i = (c - \alpha y_i)^+$, and rewrite (3.34),

$$\begin{aligned} \min_{\{y_i\}} \quad & \frac{1}{2}\sum_{i=1}^{N+1} y_i^2 + \sum_{i=1}^{N} y_i(c-\alpha y_i)^+ \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_1 \geq 0, \quad y_i \geq (c - \alpha y_{i-1})^+, \quad i = 2, \ldots, N+1, \end{aligned} \tag{3.35}$$

which is only in terms of y_i.

Next, we provide the optimal solution for the case where $y_i < \frac{c}{\alpha}$ for $i = 1, \ldots, N$. The problem in (3.35) becomes,

$$\begin{aligned} \min_{\{y_i\}} \quad & \left(\frac{1}{2} - \alpha\right)\sum_{i=1}^{N} y_i^2 + \sum_{i=1}^{N} c y_i + \frac{1}{2} y_{N+1}^2 \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_1 \geq 0, \quad y_i \geq c - \alpha y_{i-1}, \quad i = 2, \ldots, N+1. \end{aligned} \tag{3.36}$$

We write the Lagrangian for the problem in (3.36) as,

$$\begin{aligned} \mathcal{L} = & \left(\frac{1}{2} - \alpha\right)\sum_{i=1}^{N} y_i^2 + \sum_{i=1}^{N} c y_i + \frac{1}{2} y_{N+1}^2 - \lambda\left(\sum_{i=1}^{N+1} y_i - T\right) - \beta_1 y_1 \\ & - \sum_{i=2}^{N+1} \beta_i (y_i + \alpha y_{i-1} - c), \end{aligned} \tag{3.37}$$

where $\beta_i \geq 0$ and λ can be anything. The problem in (3.36) is convex since $\alpha < \frac{1}{2}$. Thus, the KKT conditions are necessary and sufficient for the optimal solution. The

KKT conditions are,

$$\frac{\partial \mathcal{L}}{\partial y_i} =(1-2\alpha)y_i + c - \lambda - \beta_i - \alpha\beta_{i+1} = 0, \quad i=1,\dots,N, \tag{3.38}$$

$$\frac{\partial \mathcal{L}}{\partial y_{N+1}} =y_{N+1} - \lambda - \beta_{N+1} = 0. \tag{3.39}$$

The complementary slackness conditions are,

$$\lambda\left(\sum_{i=1}^{N+1} y_i - T\right) = 0, \tag{3.40}$$

$$\beta_1 y_1 = 0, \tag{3.41}$$

$$\beta_i(y_i + \alpha y_{i-1} - c) = 0, \quad i=2,\dots,N+1. \tag{3.42}$$

When $y_1 > 0$ and $y_i > c - \alpha y_{i-1}$, for $i = 2,\dots,N+1$, from (3.41) and (3.42), we have $\beta_i = 0$ for all i. Then, by using (3.38) and (3.39), we have $y_i = \frac{\lambda - c}{1-2\alpha}$, for $i = 1,\dots,N$ and $y_{N+1} = \lambda$. From (3.40), we find $\lambda = \frac{(1-2\alpha)T+Nc}{N+1-2\alpha}$ which gives,

$$y_i =\frac{T-c}{N+1-2\alpha}, \quad i=1,\dots,N, \tag{3.43}$$

$$y_{N+1} =\frac{(1-2\alpha)T+Nc}{N+1-2\alpha}. \tag{3.44}$$

A typical age evolution curve is shown in Fig. 3.5(b). In order to satisfy $y_1 > 0$, $y_i > c - \alpha y_{i-1}$ for $i = 2,\dots,N+1$ and $y_i < \frac{c}{\alpha}$ for $i = 1,\dots,N$, we need $\left(\frac{N+2-\alpha}{1+\alpha}\right)c < T < \left(\frac{N+1-\alpha}{\alpha}\right)c$. Viewing this conditions in terms of T, when T is closer to the lower boundary, i.e., $\left(\frac{N+2-\alpha}{1+\alpha}\right)c < T$, we see that $y_i > c - \alpha y_{i-1}$ for $i = 2,\dots,N$ gets tighter. When T is closer to the upper boundary, we see that $y_i < \frac{c}{\alpha}$, for

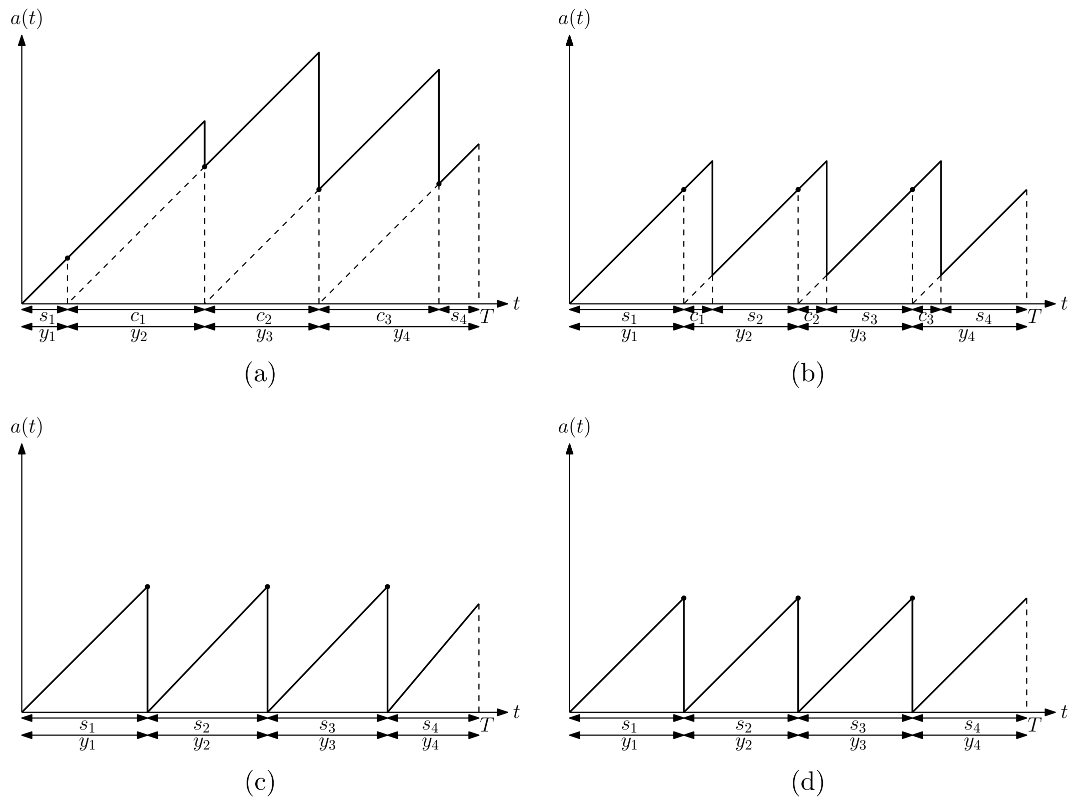


Figure 3.5: Age evolution at the receiver when the distortion function is proportional to the current age for (a) $T \leq \left(\frac{N+2-\alpha}{1+\alpha}\right)c$, (b) $\left(\frac{N+2-\alpha}{1+\alpha}\right)c < T < \left(\frac{N+1-\alpha}{\alpha}\right)c$, (c) $\frac{(N+1-\alpha)c}{\alpha} \leq T < \frac{(N+1)c}{\alpha}$, (d) $\frac{(N+1)c}{\alpha} \leq T$.

$i = 1, \dots, N$ gets tighter.

We first identify the optimal solution when $T \leq \left(\frac{N+2-\alpha}{1+\alpha}\right)c$. In the following lemma, we show that when $T \leq \left(\frac{N+2-\alpha}{1+\alpha}\right)c$, we have $y_i = c - \alpha y_{i-1}$, for $i = 2, \dots, N$.

Lemma 3.4 *In the age-optimal policy, when $T \leq \left(\frac{N+2-\alpha}{1+\alpha}\right)c$, we have $y_i = c - \alpha y_{i-1}$, for $i = 2, \dots, N$.*

Proof: We note that increasing c increases the cost of increasing y_i for $i = 1, \dots, N$ in the objective function in (3.36). Thus, increasing c yields decreasing optimal values for y_i for $i = 1, \dots, N$. We note from (3.43) that

$$\lim_{T \to \left(\frac{N+2-\alpha}{1+\alpha}\right)c} y_i = \frac{c}{1+\alpha}, \tag{3.45}$$

for $i = 1, \dots, N$. Thus, when $\left(\frac{1+\alpha}{N+2-\alpha}\right) T \leq c$, we have $y_i \leq \frac{c}{1+\alpha}$ for $i = 1, \dots, N$. Then, we have $y_i + \alpha y_{i-1} \leq c$ for $i = 2, \dots, N$. Due to the distortion constraint in the optimization problem in (3.36), we also have $y_i + \alpha y_{i-1} \geq c$ for $i = 2, \dots, N+1$. Thus, when $T \leq \left(\frac{N+2-\alpha}{1+\alpha}\right) c$, we must have $y_i = c - \alpha y_{i-1}$, for $i = 2, \dots, N$. ■

Therefore, we show in Lemma 3.4 that when $T \leq \left(\frac{N+2-\alpha}{1+\alpha}\right) c$, the optimal policy is to request the updates back-to-back except for the first update. Then, the optimal policy has the following structure,

$$y_1 = \eta, \tag{3.46}$$

$$y_i = c \sum_{j=1}^{i-1} (-\alpha)^{j-1} + (-\alpha)^{i-1} \eta, \quad i = 2, \dots, N, \tag{3.47}$$

$$y_{N+1} = T - \sum_{i=1}^{N} y_i. \tag{3.48}$$

In order to find the optimal η which minimizes the age, we substitute (3.46)-(3.48) in the objective function in (3.36), differentiate the age with respect to η, and equate to zero.

A typical age evolution curve is shown in Fig. 3.5(a). We note that when we increase c sufficiently, y_1 becomes zero. At this point, $y_1 \geq 0$ and $y_i \geq c - \alpha y_{i-1}$ for $i = 2, \dots, N$ are satisfied with equality. If we further increase c, the last feasibility constraint, $y_{N+1} \geq c - y_N$, becomes tight and the optimal solution is $y_1 = 0$,

$y_i = c - \alpha y_{i-1}$ for $i = 2, \dots, N+1$. If we increase c further, there is no feasible solution.

Next, we find the optimal solution when T is relatively large, i.e., $\left(\frac{N+1-\alpha}{\alpha}\right) c < T$. With an argument similar to that in Lemma 3.4, if c becomes smaller compared to T, the optimal value of y_i for $i = 1, \dots, N$ increases. We note that when $\lim_{T \to \left(\frac{N+1-\alpha}{\alpha}\right)c} y_i = \frac{c}{\alpha}$ for $i = 1, \dots, N$. Thus, when $\left(\frac{N+1-\alpha}{\alpha}\right) c < T$, we have $c - \alpha y_i < 0$ for $i = 1, \dots, N$. Then, the problem in (3.35) becomes,

$$\begin{aligned} \min_{\{y_i\}} \quad & \frac{1}{2} \sum_{i=1}^{N+1} y_i^2 \\ \text{s.t.} \quad & \sum_{i=1}^{N+1} y_i = T \\ & y_{N+1} \geq 0, \quad y_i \geq \frac{c}{\alpha}, \quad i = 1, \dots, N. \end{aligned} \tag{3.49}$$

We note that the problem in (3.49) is convex. Thus, the KKT conditions are necessary and sufficient for the optimal solution. After writing the KKT conditions, we observe two different optimal solution structures. When T is sufficiently large, we have $y_i > \frac{c}{\alpha}$ for all i. Then, the optimal solution is $y_i = \frac{T}{N+1}$ for all i. A typical age evolution curve is shown in Fig. 3.5(d). We need $T \geq \frac{(N+1)c}{\alpha}$ for the feasibility of the solution. When $\frac{(N+1-\alpha)c}{\alpha} \leq T < \frac{(N+1)c}{\alpha}$, we have $y_i = \frac{c}{\alpha}$ for $i = 1, \dots, N$ and $y_{N+1} = T - \sum_{i=1}^{N} y_i$. A typical age evolution curve is shown in Fig. 3.5(c).

In summary, when $T \leq \left(\frac{N+2-\alpha}{1+\alpha}\right) c$, i.e., when the total time T is small compared to the required processing time, the optimal policy is to request the updates back-to-back as shown in Fig. 3.5(a). When the total time period gets larger, the age at the receiver starts to get higher. Thus, the minimum required processing time $c - \alpha y_i$ gets smaller. Specifically, when $\left(\frac{N+2-\alpha}{1+\alpha}\right) c < T < \left(\frac{N+1-\alpha}{\alpha}\right) c$, the optimal policy is to request updates following a waiting time as shown in Fig. 3.5(b). Finally, when the age at the receiver gets even higher, i.e., when $\frac{(N+1-\alpha)c}{\alpha} \leq T$, the optimal policy is to deliver the updates without processing as shown in Figs. 3.5(c)-(d).

3.5 Numerical Results

In this section, we provide numerical results for the problems solved in Sections 3.3 and 3.4. For the numerical simulations, we use CVX tool in MATLAB [144, 145]. First, in the following subsection, we provide numerical results for the case where the maximum allowed distortion function is a constant.

3.5.1 Simulation Results for Constant Allowable Distortion

We provide five numerical results for an exponentially decaying distortion function, D_e, defined in (3.1) with $a = (1-e^{-1})^{-1}$, $b = \frac{1}{4}$ and $d = e^{-1}$. Note that we can choose the processing time c_i in $[0, 4]$. When the processing time c_i is equal to 0, the distortion function $D_e(c_i)$ attains its maximum value, i.e., $D_e(c_i) = 1$. When the processing time c_i is equal to 4, the distortion function $D_e(c_i)$ reaches its minimum value, i.e., $D_e(c_i) = 0$. Since the maximum allowed distortion is a constant, we can rewrite the distortion constraint, $D_e(c_i) \leq \beta$, as $c_i \geq c$ where $c = D_e^{-1}(\beta)$ is in $[0, 4]$. For the first four simulations, we cover each optimal policy given in Section 3.3. In these simulations, we take $T = 10$ and $N = 3$.

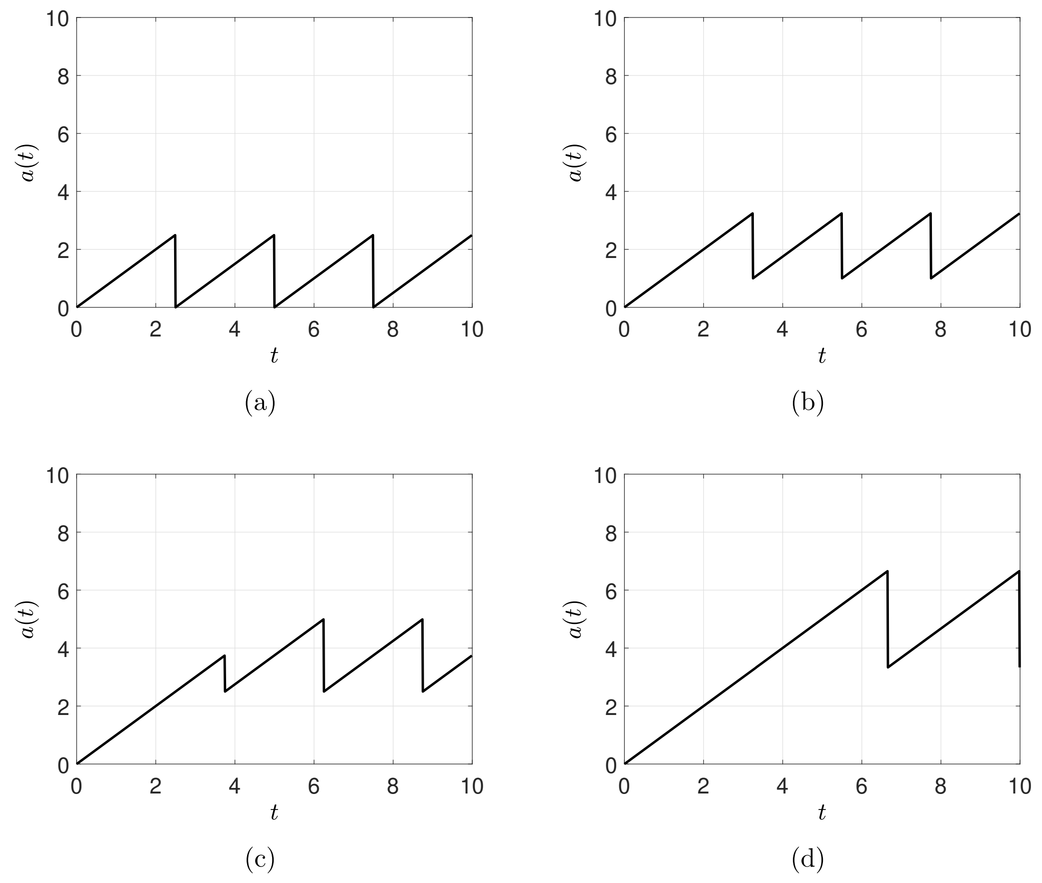


Figure 3.6: Evolution of $a(t)$ with optimal update policies for $T = 10$, $N = 3$, (a) $c = 0$, (b) $c = 1$, (c) $c = 2.5$, (d) $c = \frac{10}{3}$, when the maximum allowed distortion function is a constant.

In the first example, we take $c = 0$. In other words, there is no distortion constraint on the updates. In this case, the optimal policy is to request an update in equal time periods, i.e., $y_i = 2.5$ for all i. As there is no distortion constraint on the updates, the information provider sends the updates immediately, i.e., $c_i = 0$ for all i, and the updates have the highest possible distortion. As a result, the optimal age evolves as in Fig. 3.6(a).

In the second example, we take $c = 1$. This is the case where the minimum required processing time c is small compared to the total time duration T, i.e.,

$(N+2)c < T$. In the optimal policy, the receiver waits for an equal amount of time to request another update after the previous update is received except a longer waiting time for the first update. The optimal age evolution is given in Fig. 3.6(b). We note that the optimal policy is to request the first update after $s_1 = 2.25$ time. For the remaining updates, after the previous update is received, the receiver waits for $s_2 = s_3 = 1.25$ time to request another update. After receiving a request, the provider generates the updates after processing $c = 1$ time.

For the third example, we take $c = 2.5$. In this case, the minimum required processing time is high which means that we wish to receive the updates with lower distortion compared to previous cases. The optimal age evolution is shown in Fig. 3.6(c). We note that the optimal policy is to request the first update after waiting $s_1 = 1.25$. The receiver requests the remaining updates as soon as the previous update is received (back-to-back) since the provider uses relatively large amount of time to generate updates. In this case, the provider processes each update for $c_i = 2.5$ time for all i.

For the fourth example, we take $c = \frac{10}{3}$ which is the highest possible minimum required processing time as $Nc = T$. In this case, there is only one feasible solution, which is to request the first update at $t = 0$ and the remaining updates as soon as the previous update is received (back-to-back), i.e., $s_i = 0$ for all i. The provider processes each update for $c_i = \frac{10}{3}$ time for all i. The optimal age evolves as in Fig. 3.6(d).

Finally, we note that there is a trade-off between age and distortion. If we increase the distortion constraint β (hence decrease the processing time constraint

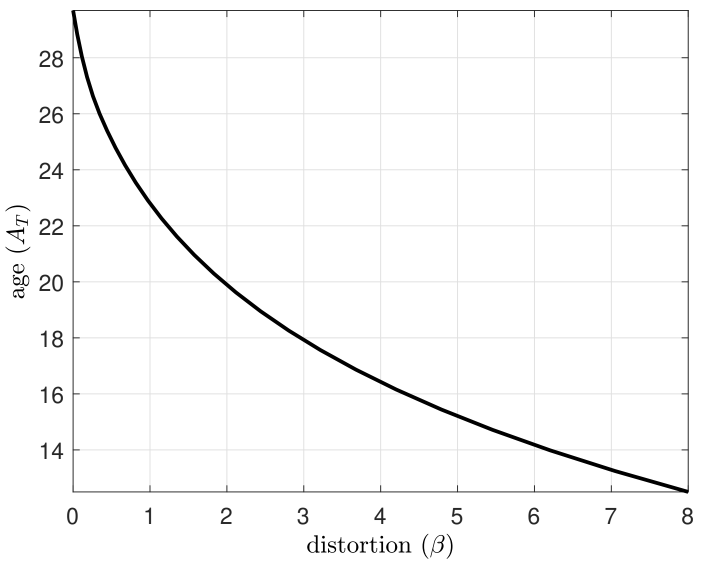


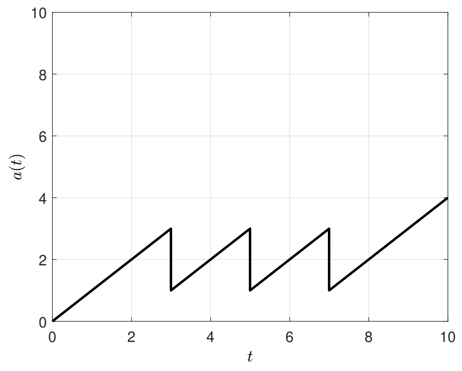
Figure 3.7: Age versus distortion of the updates for $a = \frac{8}{1-e^{-3}}$, $b = 1.2$, and $d = e^{-3}$ in (3.1) when the maximum allowed distortion is a constant. We vary β and find the minimum age for each β.

c), then we achieve a lower average age at the receiver, but the receiver obtains updates with low quality as the distortion of the updates is high. On the other hand, if we decrease the distortion constraint β (hence increase the processing time constraint c), the receiver obtains updates with high quality, but in this case, the average age at the receiver increases. We show this trade-off between age and distortion as a fifth example in Fig. 3.7.

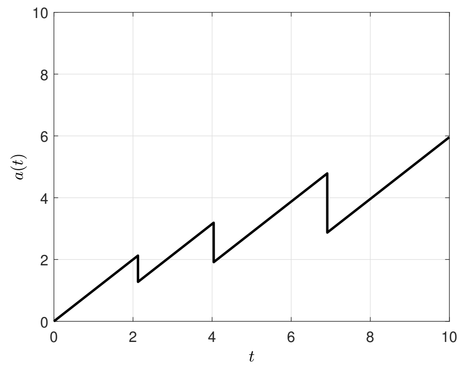
Next, in the following subsection we provide numerical results for the case where the maximum allowed distortion function depends on the current age.

3.5.2 Simulation Results for Age-Dependent Allowable Distortion

First, we provide three numerical results for the case where the maximum allowed distortion function is inversely proportional to the instantaneous age, i.e., we have



(a)



(b)

Figure 3.8: Evolution of $a(t)$ with optimal update policies for $T = 10$, $N = 3$, (a) $\alpha = 0.5$, (b) $\alpha = 1.5$, when the maximum allowed distortion function is inversely proportional to the current age, i.e., $c_i \geq \alpha y_i$.

$c_i \geq \alpha y_i$ constraint for each update.

For the first example, we take $T = 10$, $N = 3$ and $\alpha = 0.5$. This example corresponds to the case where the maximum allowed distortion slowly decreases with the current age, i.e., α is small. The optimal solution follows (3.28) and (3.29) and is equal to $y_i = 2$ for $i = 1, 2, 3$ and $y_4 = 4$. We note that the information receiver requests all the updates when its age is equal to $y_i = 2$, and then, lets its age grow for the remaining time. Since $c_i = \alpha y_i$, we have $c_i = 1$ for all i which means that all the updates have the same level of distortion as the processing times for the updates are equal. We observe in Fig. 3.8(a) that the optimal policy resembles the optimal policy for the case with constant allowable distortion when the minimum required processing time is small, i.e., the second example shown in Fig. 3.6(b) in Section 3.5.1.

For the second example, we take $T = 10$, $N = 3$ and $\alpha = 1.5$. This example corresponds to the case where the maximum allowed distortion decreases faster

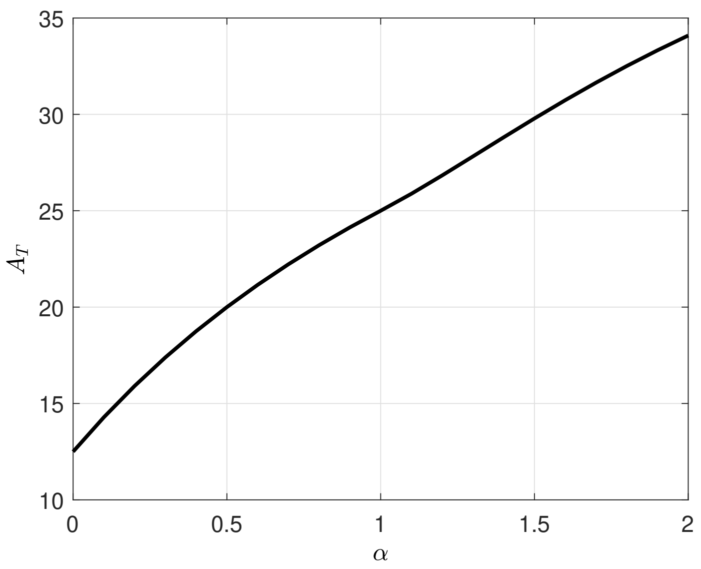


Figure 3.9: Average age versus α for $T = 10$ and $N = 3$ when the maximum allowed distortion function is inversely proportional to the current age, i.e., $c_i \geq \alpha y_i$. We vary α in between 0 and 2 and find the corresponding minimum age for each α.

with the instantaneous age, i.e., α is large. The optimal policy follows (3.31)-(3.33) and the optimal age evolution is shown in Fig. 3.8(b). The optimal solution is $y_1 = 0.8511$, $y_2 = 1.2766$, $y_3 = 1.9149$ and $y_4 = 5.9574$. Due to $c_i = \alpha y_i$, we have $c_1 = 1.2766$, $c_2 = 1.9149$ and $c_3 = 2.8723$. We observe different from the first example where $\alpha < 1$ that the processing time for each update is different which also means that updates have different levels of distortion. We also note that updates are requested right after the previous update is received except for the first update, i.e., $s_i = 0$ for $i = 2, \dots, N$.

For the third example, we take $T = 10$, $N = 3$ and vary α in between 0 and 2. When α gets larger, the receiver starts to require updates with lower distortion. In other words, with a larger α, the transmitter allocates more time for processing the updates which increases the average age of the receiver as shown in Fig. 3.9.

When the maximum allowed distortion function is inversely proportional to the age, two different optimum policies are observed depending on the value of α as shown in Figs. 3.8(a)-(b). Due to these two different update policies, we observe two different monotonically increasing functions with respect to α in Fig. 3.9, i.e., one is in between $\alpha \in [0, 1]$ and the other one is in between $\alpha \in [1, 2]$.

In the following five examples, we consider the case where the maximum allowed distortion function is proportional to the current age, i.e., we have $c_i \geq c - \alpha y_i$ constraint for each update. We take $N = 3$, $c = 1$, $\alpha = 0.4$.

For the first example, we take $T = 3$ which corresponds to the case where T is relatively small compared to the minimum required processing time. The optimal policy follows (3.46)-(3.48). The optimal solution is to choose $y_1 = 0.4478$, $y_2 = 0.8209$, $y_3 = 0.6716$ and $y_4 = 1.0597$. Since $c_i = c - \alpha y_i$, we have $c_1 = 0.8209$, $c_2 = 0.6716$ and $c_3 = 0.7313$. The optimal age evolution is shown in Fig. 3.10(a). We observe that updates are requested right after the previous update is received except for the first update, i.e., $s_i = 0$ for $i = 2, \dots, N$. In this case, as the instantaneous age is relatively low when the update is requested, the information provider processes the updates further to generate updates with high quality.

For the second example, we take $T = 6$ which corresponds to the case where T is relatively large compared to the minimum required processing time. The optimal solution follows (3.43)-(3.44) and $y_1 = y_2 = y_3 = 1.5625$ and $y_4 = 1.3125$. We have $c_i = 0.3750$ for all i. The optimal age evolution is shown in Fig. 3.10(b). As the instantaneous age is higher when the updates are requested compared to the first example, the system imposes a low distortion constraint for each update. We

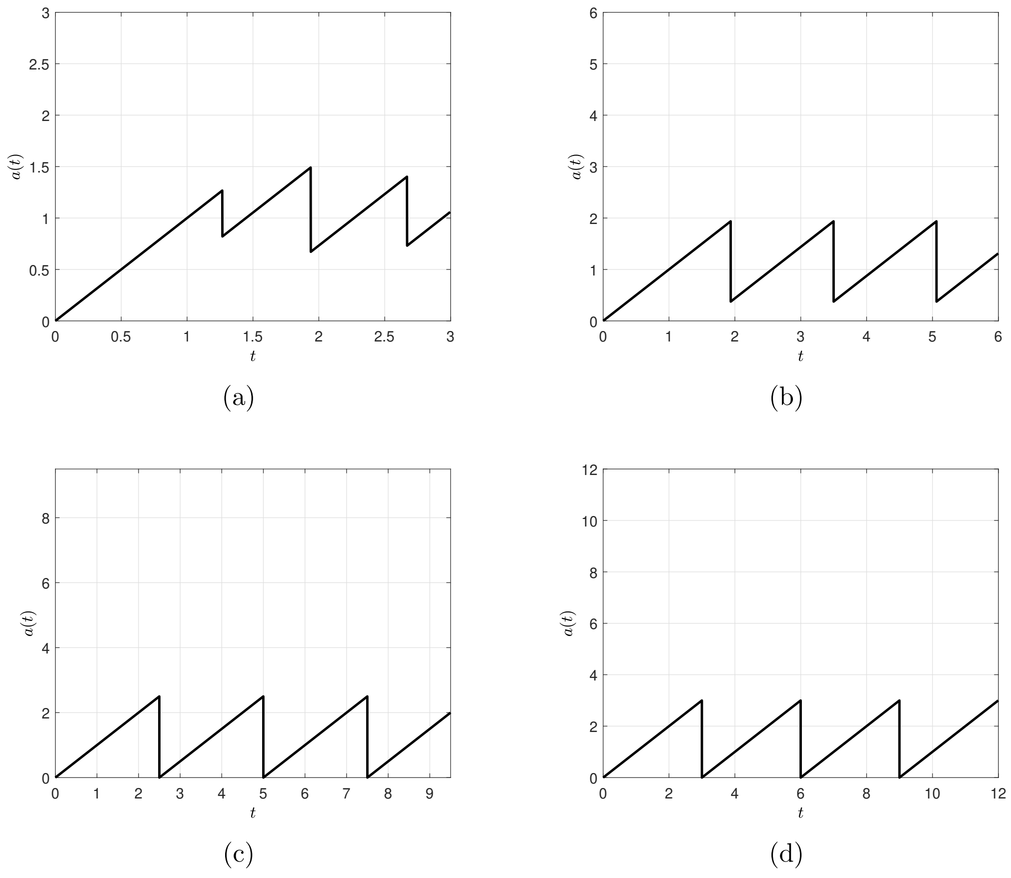


Figure 3.10: Evolution of $a(t)$ with optimal update policies for $c = 1$, $N = 3$, $\alpha = 0.4$, (a) $T = 3$, (b) $T = 6$, (c) $T = 9.5$, (d) $T = 12$, when the maximum allowed distortion function is an increasing function of the current age, i.e., $c_i \geq c - \alpha y_i$.

observe that as the receiver requests all the updates when the age at the receiver is equal to $y_i = 1.5625$ for $i = 1, 2, 3$, the distortion constraint for each update becomes the same.

For the third example, we take $T = 9.5$ which corresponds to the case where the optimal policy follows $y_i = \frac{c}{\alpha}$ and $y_{N+1} = T - \sum_{i=1}^{N} y_i$. The optimal solution is $y_i = 2.5$ for $i = 1, 2, 3$ and $y_4 = 2$. In this case, as the instantaneous age gets higher when the update is requested, freshness of the updates becomes more important than the quality of the updates. That is why in this case, there is no active distortion constraints on the updates, i.e., $c_i \geq 0$. Thus, the receiver sends the updates without any processing, i.e., $c_i = 0$ for all i. The optimal age evolution is shown in Fig. 3.10(c). Since the processing time for each update is equal to zero, the updates are not aged during the processing time and the age of the receiver reduces to zero after receiving each update.

For the fourth example, we take $T = 12$. The optimal policy follows $y_i = \frac{T}{N+1}$ and is equal to $y_i = 3$ and $c_i = 0$ for all i. The optimal age evolution is shown in Fig. 3.10(d). In this case, we observe a similar optimal solution structure as in the previous case where $T = 9.5$. As the updates are requested when the age is too high, updates with the highest distortion become acceptable for the system. We thus observe the same optimal solution structure as in the case with constant allowable distortion when there is no active distortion constraint, i.e., when $c = 0$ in the first example shown in Fig. 3.6(a) in Section 3.5.1.

For the fifth example, we take $T = 4$, $N = 3$ and vary α in between 0 and 0.49. When α gets larger, the receiver starts to accept updates with higher distortion which decreases the minimum required processing time. That is why the minimum average age decreases with α as shown in Fig. 3.11. We note that when $\alpha \in [0, 0.2]$, the optimal policy follows the model shown in Fig. 3.10(a). When $\alpha \in (0.2, 0.49]$, the optimal policy follows the model shown in Fig. 3.10(b). Due to these two different update policies, we observe two different monotonically decreasing functions of α connected at $\alpha = 0.2$ in Fig. 3.11.

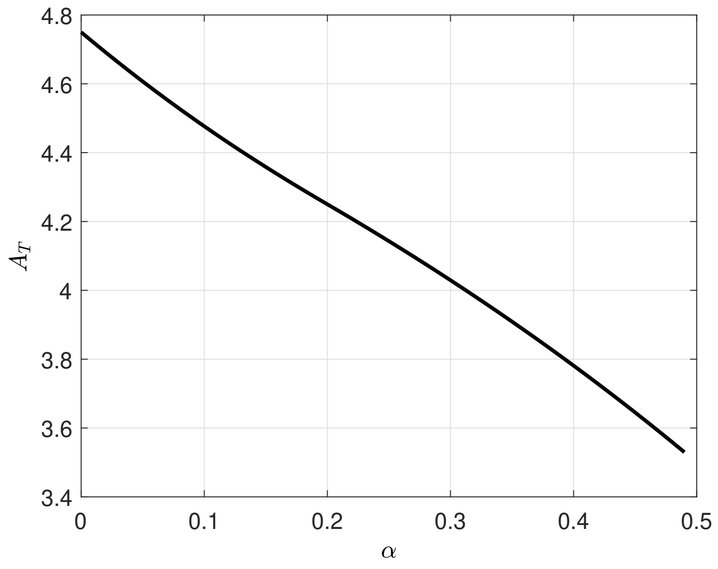


Figure 3.11: Average age versus α for $T = 4$ and $N = 3$ when the maximum allowed distortion function is proportional to age, i.e., $c_i \geq c - \alpha y_i$. We vary $\alpha \in [0, 0.49]$ and find the corresponding minimum age for each α.

3.6 Conclusion

In this chapter, we considered the concept of status updating with update packets subject to distortion. In this model, updates are generated at the information provider (transmitter) following an update generation process that involves collecting data and performing computations. The distortion in each update decreases with the processing time during update generation at the transmitter; while processing longer generates a better-precision update, the long processing time increases the age of information. This implies that there is a trade-off between precision (quality) of information and age (freshness) of information. The system may be designed to strike a desired balance between quality and freshness of information. In this chapter, we determined this design, by solving for the optimum update scheme subject to a desired distortion level.

We considered the case where the maximum allowed distortion does not depend on the current age, i.e., is a constant, and the case where the maximum allowed distortion depends on the current age. For this case, we considered two sub-cases, where the maximum allowed distortion is a decreasing function and an increasing function of the current age.

We note that while we formulated the allowable distortion constraint using the *current age at the receiver*, we could similarly formulate it by using *time elapsed since the last requested update*. Specifically, we could use the constraint $c_i \geq \alpha s_i$ instead of the constraint $c_i \geq \alpha y_i$ in (3.20) and the constraint $c_i \geq c - \alpha s_i$ instead of the constraint $c_i \geq c - \alpha y_i$ in (3.34). We note that these two considerations are similar: If the receiver has not requested an update for a long time (large s_i), its current age will be high (large y_i). In order to avoid repetitive arguments, in this chapter, we only considered the case where the distortion constraint depends on the instantaneous age y_i at the receiver at the time of requesting a new update.

CHAPTER 4

Source Coding for Age of Information

4.1 Introduction

In this chapter, we consider a status updating system that consists of a single transmitter node and a single receiver node (see Fig. 4.1). The transmitter receives independent and identically distributed time-sensitive status update packets generated by an information source based on an observed random phenomenon that takes n distinct values with a known probability mass function (pmf). This observed random variable could be the position of a UAV in autonomous systems or share prices in the stock market. Arriving status update packets are encoded at the transmitter and sent to the receiver through an error-free noiseless channel. The receiver wants to acquire fresh information regarding the observed random variable, which brings up the concept of age of information.

Unlike most of the literature in which the transmission times, also referred to as service times in queueing theory, are based on a given service distribution, in this work, we design transmission times through source coding schemes by choosing the codeword lengths assigned to realizations. That is, the codeword length assigned to

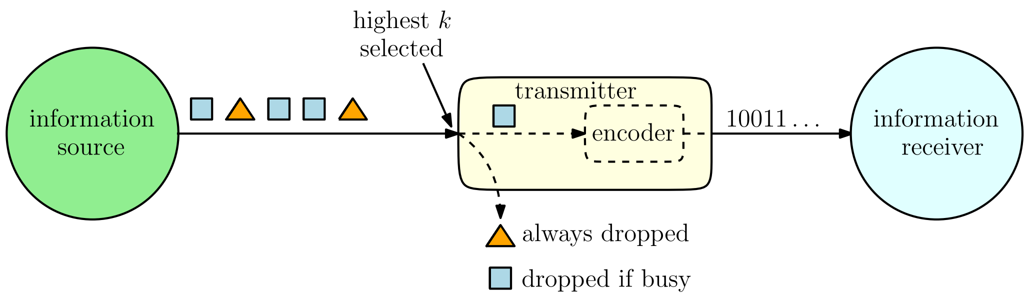


Figure 4.1: An information source generates i.i.d. status updates from a random variable X. Only a portion of the realizations (shown with a square) is encoded into codewords. Update packets that come from the selected portion of the realizations that find the transmitter node idle are sent to the receiver node. Non-selected realizations (shown with a triangle) are always discarded at the transmitter node even if the transmitter node is idle.

each realization represents the service time (transmission time) of that realization.

References that are most closely related to our work are [77, 78, 80] which study the timely source coding problem for a discrete-time system. References [80] and [77] study a communication system where a source follows a zero-wait update generation model whereas in [78] status updates arrive exogenously as a Bernoulli process with probability q. In [80], the transmitter only skips the status updates that are generated while the channel is busy. Unlike the model in [80], references [77] and [78] reconstruct the entire source message stream in a lossless manner. Reference [80] finds real-relaxation of the underlying integer-valued codeword lengths using Shannon codes based on a modified version of the given pmf that achieve the optimal age with a constant gap. References [77] and [78] consider block coding and source coding problems to find age-optimal codes for FIFO queues.

Different from [77, 78, 80], in our model, time is not slotted and the status update packets arrive at the transmitter node following a Poisson process with a known parameter λ. Unlike the model in [80], we introduce an encoding mechanism where the transmitter skips not only the status updates that are generated while the channel is busy but also the least probable ones to achieve a lower average age of information at the receiver. We term this encoding mechanism *selective encoding.* In this selective encoding model, instead of encoding all possible realizations, we encode only a portion of the realizations and send to the receiver node. Specifically, we consider what we call the *highest k selective encoding* scheme in which we only encode the most probable k realizations and disregard any update packets from the remaining $n-k$ realizations. We note that a smaller k yields shorter codeword lengths but larger interarrival times, as in this case most of the updates are not encoded. However, when k is large, codeword lengths and correspondingly the transmission times get larger even though the interarrival times get smaller. Thus, in this chapter, based on the given pmf, we aim to find the optimal k which strikes a balance between these two opposing trends such that the average age at the receiver node is minimized. Due to this selective encoding scheme not every realization is sent to the receiver even if the channel is free.

Next, we consider a scenario in which the remaining $n-k$ realizations are not completely disregarded but encoded with a certain probability which we call the *randomized selective encoding* scheme. In this scheme, in addition to the most probable k realizations, the remaining $n-k$ less probable realizations are sometimes encoded.

A disadvantage of the highest k selective encoding scheme is the fact that the receiver node is not informed when one of the non-selected realizations occurs. For instance, during a period with no arrivals, the receiver node cannot differentiate

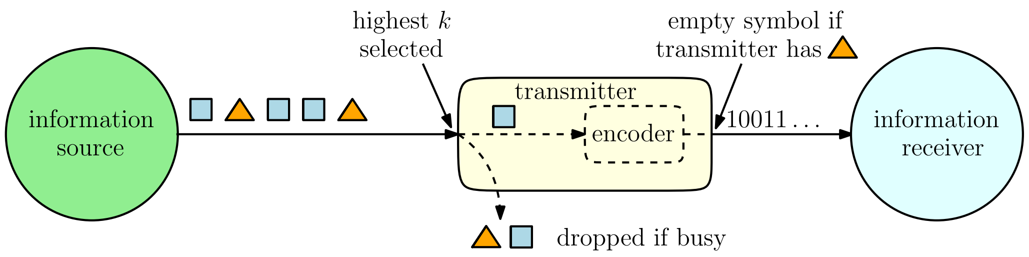


Figure 4.2: Update packets that come from the selected portion of the realizations (shown with a square) that find the transmitter idle are sent to the receiver. Non-selected realizations (shown with a triangle) that find the transmitter idle are mapped into an empty symbol.

whether there has been no arrivals or if the arrival has taken one of the non-selected values as in either case it does not receive any update packets. Thus, lastly, we take a careful look at the remaining $n-k$ realizations and propose a modified selective encoding policy which we call the *highest k selective encoding with empty symbol* that still achieves a lower average age than encoding every realization but also informs the receiver node when one of the non-selected values is taken by the observed random variable. In this scheme, only the most probable k realizations are encoded and the remaining $n-k$ realizations are mapped into a designated empty symbol such that in the case of these $n-k$ non-selected realizations, this empty symbol is sent to further inform the receiver (see Fig. 4.2). Thus, in such a case, the receiver at least knows that the observed random variable has taken a value from the non-selected portion even though it does not know which value was taken specifically. We consider two variations on this policy: when the empty symbol does not reset the age and when the empty symbol resets the age.[1]

[1]When the empty symbol x_e is received, the receiver does not know exactly which update is realized at the source. For that reason, operationally, the receiver may not reset its age when x_e is received. On the other hand, the receiver may reset its age as the empty symbol carries some *partial* information regarding the status update at the source, i.e., when the empty symbol x_e is received, the receiver knows that one of the least probable status updates is realized at the source. That is why we consider both of these scenarios in our analysis.

For all three encoding schemes, we find the average age experienced by the receiver node and determine the age-optimal real codeword lengths, including the codeword length of the empty symbol in the case of the highest k selective encoding with empty symbol scheme. Through numerical evaluations for given arbitrary pmfs, we show that the proposed selective encoding policies achieve a lower average age than encoding every realization, and find the corresponding age-optimal k values. In addition, we discuss the optimality of the highest k selective encoding policy. We note that, since we focus on age-optimal real-valued codeword lengths in this chapter, the obtained age values serve as lower bounds to what can be attained by integer-valued codeword lengths. Designing age-optimal integer-valued codeword lengths is not addressed in this chapter and remains an interesting open problem.

Finally, a similar k out of n type of idea was used in [19–22, 24] in the context of multicasting updates in networks, where each packet is transmitted until the earliest k out of n receiver nodes have received the packet. While the multicast communication problem studied in [19–22,24] and the source coding problem studied here are fundamentally different, there is an analogy between their results as follows. In [19–22, 24], it was shown that sending status updates to (earliest) k out of n receivers achieves a smaller average age of information than sending status updates to every one of n receivers. Analogously, in this chapter, we show that sending status updates for (most probable) k out of n realizations achieves a smaller age of information than sending status updates for every one of n realizations.

4.2 System Model and Problem Formulation

We consider a communication system in which an information source generates independent and identically distributed status update packets from an observed phenomenon that takes realizations from the set $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ based on a known pmf $P_X(x_i)$ for $i \in \{1, \dots, n\}$.[2] Without loss of generality, we assume that $P_X(x_m) \geq P_X(x_{m+1})$ for all m, i.e., the probabilities of the realizations are in a non-increasing order. Update packets arrive at the transmitter node following a Poisson process with parameter λ. The transmitter node implements a blocking policy in which the update packets that arrive when the transmitter node is busy are blocked and lost. Thus, the transmitter node receives only the updates that arrive when it is idle.

We consider three different encoding policies: highest k selective encoding, randomized selective encoding, and highest k selective encoding with an empty symbol.

4.2.1 Policy 1: Highest k Selective Encoding

In the first policy, we consider a selective encoding mechanism that we call *highest k selective encoding* where the transmitter node only sends the most probable k realizations, i.e., only the realizations from set $\mathcal{X}_k = \{x_1, \dots, x_k\}$, which have

[2]Even though the number of realizations, n, is a finite number, it can be arbitrarily large. We highlight that this modeling choice is well suited to many practical applications even when the underlying randomness is continuous so long as it is quantized to sufficiently many discrete levels, e.g., quantized temperature sensor measurements, trajectory commands in a UAV system such as move right, left, up, down, and so on.

the highest probabilities among possible n updates generated by the source, are transmitted for $k \in \{1, \ldots, n\}$; see Fig. 4.1. If an update packet from the remaining non-selected portion of the realizations arrives, the transmitter disregards that update packet and waits for the next arrival. If an update packet arrives from the selected portion of the realizations, then the transmitter encodes that update packet by using a binary alphabet with the conditional probabilities given by,

$$P_{X_k}(x_i) = \begin{cases} \frac{P_X(x_i)}{q_k}, & i = 1, 2, \ldots, k \\ \\ 0, & i = k+1, k+2, \ldots, n, \end{cases} \tag{4.1}$$

where

$$q_k \triangleq \sum_{\ell=1}^{k} P_X(x_\ell). \tag{4.2}$$

The transmitter assigns codeword $c(x_i)$ with length $\ell(x_i)$ to realization x_i for $i \in \{1, 2, \ldots, k\}$.

4.2.2 Policy 2: Randomized Selective Encoding

In the second policy, inspired by [80], we study a *randomized selective encoding* scheme. In this policy, the most probable k realizations are always encoded. However, instead of discarding the remaining $n-k$ realizations, the transmitter node encodes them with probability α and discards them with probability $1-\alpha$. In other words, in this model, less likely realizations that are not encoded under the highest

k selective encoding policy are sometimes transmitted to the receiver node. Thus, under this operation, codewords for each one of the n possible realizations need to be generated since every realization can be sent to the receiver node. The transmitter assigns codeword $c(x_i)$ with length $\ell(x_i)$ to realization x_i for $i \in \{1, 2, \dots, n\}$.

The transmitter node performs encoding using the following conditional probabilities,

$$P_{X_\alpha}(x_i) = \begin{cases} \frac{P_X(x_i)}{q_{k,\alpha}}, & i = 1, 2, \dots, k \\ \\ \alpha\frac{P_X(x_i)}{q_{k,\alpha}}, & i = k+1, k+2, \dots, n, \end{cases} \tag{4.3}$$

where

$$q_{k,\alpha} \triangleq \sum_{\ell=1}^{k} P_X(x_\ell) + \alpha \sum_{\ell=k+1}^{n} P_X(x_\ell). \tag{4.4}$$

4.2.3 Policy 3: Highest k Selective Encoding with an Empty Symbol

In the third policy, we consider an encoding scheme that we call the *highest k selective encoding with an empty symbol.* In this encoding scheme, the transmitter always encodes the most probable k realizations as in the previous two policies. However, unlike the previous models, if an update packet from the remaining non-selected portion of the realizations arrives, the transmitter sends an empty status update denoted by x_e to further inform the receiver at the expense of longer codewords for the selected k realizations.

When an update packet arrives from the set $\mathcal{X}'_k = \mathcal{X}_k \cup \{x_e\}$, the transmitter node encodes that update packet with the binary alphabet by using the pmf given as $\{P_X(x_1), P_X(x_2), \dots, P_X(x_k), P_X(x_e)\}$ where $P_X(x_e) = 1 - q_k$. Thus, in this policy, the transmitter node assigns codewords to the most probable k realizations as well as to the empty symbol x_e. That is, the transmitter assigns codeword $c(x_i)$ with length $\ell(x_i)$ to realization x_i for $i \in \{1, \dots, k, e\}$.

In this chapter, we focus on the source coding aspect of timely status updating. Therefore, in all these three policies, the channel between the transmitter node and the receiver node is error-free. The transmitter node sends one bit at a unit time. Thus, if the transmitter node sends update x_i to the receiver node, this transmission takes $\ell(x_i)$ units of time. That is, for realization x_i, the service time of the system is $\ell(x_i)$.

4.2.4 Problem Formulation

We use the age of information metric to measure the freshness of the information at the receiver node. Let $\Delta(t)$ be the instantaneous age at the receiver node at time t with $\Delta(0) = \Delta_0$. Age at the receiver node increases linearly in time and drops to the age of the most recently received update upon delivery of a new update packet. We define the long term average age as,

$$\Delta = \lim_{T \to \infty} \frac{1}{T} \int_0^T \Delta(t) dt. \tag{4.5}$$

Our aim is to find the codeword lengths for each encoding policy described in Sections 4.2.1, 4.2.2, and 4.2.3 that minimize the long term average age for a given

k such that a uniquely decodable code can be designed, i.e., the Kraft inequality is satisfied [146].

In the following section, we find an analytical expression for the long term average age Δ.

4.3 Average Age Analysis

As described in Section 4.2, status update packets arrive at the transmitter as a Poisson process with rate λ. Update packets that arrive when the transmitter is busy are blocked from entry and dropped. Thus, upon successful delivery of a packet to the receiver, the transmitter idles until the next update packet arrives. This idle waiting period in between two arrivals is denoted by Z which is an exponential random variable with rate λ due to the memoryless property of exponential random variables as update interarrivals at the transmitter are exponential with λ.

We note that in all of the encoding policies in Section 4.2, every packet from the set $\mathcal{X}_k$ which successfully enters the transmitter node is always sent to the receiver. However, a packet from the remaining least probable $n-k$ realizations which enters the transmitter might not be sent. Under the highest k selective encoding policy described in Section 4.2.1, when one of the remaining $n-k$ packets enters the transmitter node, the transmitter node drops the packet and proceeds to wait for the next update arrival. Under the randomized selective encoding scheme described in Section 4.2.2, remaining $n-k$ less likely realizations are transmitted to the receiver node with probability α. Under the highest k selective encoding scheme with an empty symbol described in Section 4.2.3, the transmitter node sends a designated empty status update to further inform the receiver about the occurrence of a realization from the remaining $n-k$ realizations.

We denote the update packets which arrive when the transmitter node is idle and reset the age as *successful* update packets. Since the channel is noiseless and there is no preemption, these successful packets are received by the receiver node. We denote T_{j-1} as the time instant at which the jth successful update packet is received. We define update cycle denoted by $Y_j = T_j - T_{j-1}$ as the time in between two successive successful update arrivals at the transmitter. Update cycle Y_j consists of a busy cycle and an idle cycle such that

$$Y_j = S_j + W_j, \tag{4.6}$$

where S_j is the service time of update j and W_j is the overall waiting time in the jth update cycle.[3]

Fig. 4.3 shows a sample age evolution at the receiver. Here, Q_j denotes the area under the instantaneous age curve in update cycle j and Y_j denotes the length of the jth update cycle as defined earlier. The metric we use, long term average age, is the average area under the age curve which is given by [10]

$$\Delta = \limsup_{n\to\infty} \frac{\frac{1}{n}\sum_{j=1}^{n} Q_j}{\frac{1}{n}\sum_{j=1}^{n} Y_j} = \frac{\mathbb{E}[Q]}{\mathbb{E}[Y]}. \tag{4.7}$$

[3]We note that W_j can be thought of as the idle time following the service completion of update j.

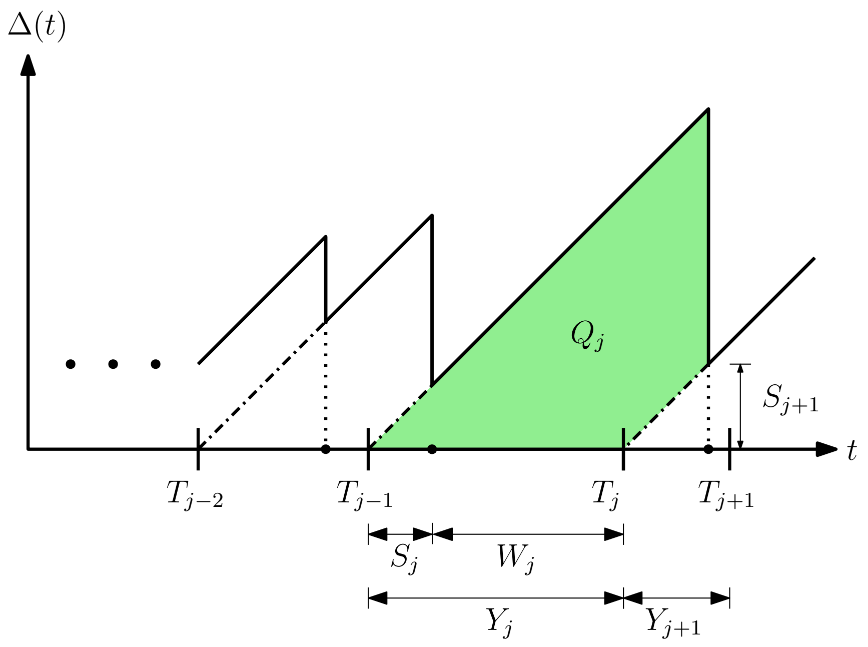


Figure 4.3: Sample age evolution $\Delta(t)$ at the receiver node. Successful updates are indexed by j. The jth successful update arrives at the server node at T_{j-1}. Update cycle at the server node is the time in between two successive arrivals and is equal to $Y_j = S_j + W_j = T_j - T_{j-1}$.

By using Fig. 4.3, we find $Q_j = \frac{1}{2}Y_j^2 + Y_jS_{j+1}$, where Y_j is given in (4.6). Thus, using the independence of Y_j and S_{j+1}, (4.7) is equivalent to

$$\Delta = \frac{\mathbb{E}[Y^2]}{2\mathbb{E}[Y]} + \mathbb{E}[S]. \tag{4.8}$$

In the following section, we find the optimal real-valued codeword lengths for the highest k selective encoding policy described in Section 4.2.1.

4.4 Optimal Codeword Design Under Selective Encoding

In this section, we consider the highest k selective encoding policy described in Section 4.2.1. Under this way of operation, the transmitter only sends the most probable k realizations from the set $\mathcal{X}_k$, and drops any update packets from the remaining $n-k$ least probable realizations.

Proposition 4.1 characterizes the average age Δ given in (4.8) for the encoding scheme described in Section 4.2.1.

Proposition 4.1 *Under the highest k selective encoding scheme, the average age at the receiver node is given by*

$$\Delta = \frac{\mathbb{E}[L^2] + \frac{2}{q_k\lambda}\mathbb{E}[L] + \frac{2}{(q_k\lambda)^2}}{2\left(\mathbb{E}[L] + \frac{1}{q_k\lambda}\right)} + \mathbb{E}[L], \tag{4.9}$$

where the first and the second moments of the codeword lengths are given by $\mathbb{E}[L] = \sum_{i=1}^{k} P_{X_k}(x_i)\ell(x_i)$ and $\mathbb{E}[L^2] = \sum_{i=1}^{k} P_{X_k}(x_i)\ell(x_i)^2$.

Proof: With the highest k selective encoding scheme, we note that the overall waiting time W is equal to $W = \sum_{\ell=1}^{M} Z_\ell$ where Z_ℓs are the i.i.d. exponential random variables with rate λ as discussed earlier. Here, M is a geometric random variable with parameter q_k (defined in (4.2)) which denotes the total number of update arrivals until the first update from the set $\mathcal{X}_k$ is observed at the transmitter node. W is also an exponential random variable with rate λq_k [147, Prob. 9.4.1]. Then, noting that the service time random variable S in (4.6) is the codeword length random variable L, we have

$$\mathbb{E}[Y] = \mathbb{E}[L] + \mathbb{E}[W], \tag{4.10}$$

$$\mathbb{E}[Y^2] = \mathbb{E}[L^2] + 2\mathbb{E}[W]\mathbb{E}[L] + \mathbb{E}[W^2], \tag{4.11}$$

where $\mathbb{E}[W] = \frac{1}{q_k\lambda}$ and $\mathbb{E}[W^2] = \frac{2}{(q_k\lambda)^2}$. Substituting (4.10) and (4.11) in (4.8) yields the result in (4.9).[4] ■

Thus, (4.9) characterizes the average age Δ achieved at the receiver node in terms of the first and second moments of the codeword lengths for a given pmf, selected k, and update arrival rate λ. Next, we formulate the age minimization problem as,

$$\begin{aligned} \min_{\{\ell(x_i)\}} \quad & \frac{\mathbb{E}[L^2] + 2a\mathbb{E}[L] + 2a^2}{2(\mathbb{E}[L]+a)} + \mathbb{E}[L] \\ \text{s.t.} \quad & \sum_{i=1}^{k} 2^{-\ell(x_i)} \leq 1 \\ & \ell(x_i) \in \mathbb{R}^+, \quad i \in \{1,\dots,k\}, \end{aligned} \tag{4.12}$$

where the objective function is equal to the average age found in Proposition 4.1 with $a = \frac{1}{\lambda q_k}$, the first constraint is the Kraft inequality, and the second constraint represents the feasibility of the codeword lengths, i.e., each codeword length should be non-negative.

We note that the optimization problem in (4.12) is a nonlinear fractional problem. To solve this problem, we use the fractional programming method introduced in [148].[5] For that, we define the following intermediate problem, which is parame-

[4]Under the assumption that $\mathbb{E}[L]$ and $\mathbb{E}[L^2]$ exist and are finite, we note that the average AoI expression in (4.9) is aligned with [10, Theorem 1], as expected.

[5]Here, we note that in order to apply fractional programming method, the feasible set for $\ell(x_i)$ needs to be compact in (4.12), i.e., $0 \leq \ell(x_i) \leq C$ for all $x_i \in \mathcal{X}_k$ where C is the upper bound on the codeword lengths. By assuming that $P_{X_k}(x_i) > 0$ for the encoded updates, we can always choose the upper bound on $\ell(x_i)$ arbitrarily large, and thus we can implement this technique.

terized by θ, similar to [31] and [46]

$$\begin{aligned} p(\theta) := \min_{\{\ell(x_i)\}} \ & \frac{1}{2}\mathbb{E}[L^2] + \mathbb{E}[L]^2 + (2a-\theta)\mathbb{E}[L] + a^2 - \theta a \\ \text{s.t.} \quad & \sum_{i=1}^{k} 2^{-\ell(x_i)} \leq 1 \\ & \ell(x_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, k\}. \end{aligned} \tag{4.13}$$

One can show that $p(\theta)$ is decreasing in θ and the optimal solution is obtained when $p(\theta) = 0$ such that the optimal age for the problem in (4.12) is equal to θ, i.e., $\Delta^* = \theta$ [148]. We define the Lagrangian [149] function for (4.13) as

$$\mathcal{L} = \frac{1}{2}\mathbb{E}[L^2] + \mathbb{E}[L]^2 + (2a-\theta)\mathbb{E}[L] + a^2 - \theta a + \beta\left(\sum_{i=1}^{k} 2^{-\ell(x_i)} - 1\right), \tag{4.14}$$

where $\beta \geq 0$. Next, we write the KKT conditions as

$$\frac{\partial \mathcal{L}}{\partial \ell(x_i)} = P_{X_k}(x_i)\ell(x_i) + 2\mathbb{E}[L]P_{X_k}(x_i) + (2a-\theta)P_{X_k}(x_i) - \beta(\log 2)2^{-\ell(x_i)} = 0, \quad \forall i, \tag{4.15}$$

and the complementary slackness condition as

$$\beta\left(\sum_{i=1}^{k} 2^{-\ell(x_i)} - 1\right) = 0. \tag{4.16}$$

In the following lemma, we prove that the optimal codeword lengths must satisfy the Kraft inequality as an equality.

Lemma 4.1 *For the age-optimal real codeword lengths, we must have* $\sum_{i=1}^{k} 2^{-\ell(x_i)}=1$.

Proof: Assume that the optimal codeword lengths satisfy $\sum_{i=1}^{k} 2^{-\ell(x_i)} < 1$, which implies that $\beta = 0$ due to (4.16). From (4.15), we have

$$P_{X_k}(x_i)\ell(x_i) + 2\left(\sum_{j=1}^{k} P_{X_k}(x_j)\ell(x_j)\right) P_{X_k}(x_i) + (2a-\theta)P_{X_k}(x_i) = 0, \quad \forall i. \quad (4.17)$$

Summing (4.17) over all $i \in \{1, \dots, k\}$ we find

$$3\mathbb{E}[L] + 2a - \theta = 0, \quad (4.18)$$

where $\mathbb{E}[L]$ is as in Proposition 4.1. Thus, we find $\ell(x_i) = \frac{\theta-2a}{3}$ for all $i \in \{1, 2, \dots, k\}$. Thus, $\mathbb{E}[L] = \frac{\theta-2a}{3}$ and $\mathbb{E}[L^2] = \left(\frac{\theta-2a}{3}\right)^2$ so that $p(\theta) = -\frac{\theta^2}{6} - \frac{\theta a}{3} + \frac{a^2}{3}$. By using $p(\theta) = 0$, we find $\theta = (-1+\sqrt{3})a$ which gives $\ell(x_i) = \frac{(-3+\sqrt{3})a}{3} < 0$ for $i \in \{1, 2, \dots, k\}$. Since the codeword lengths cannot be negative, we reach a contradiction. Thus, the optimal codeword lengths must satisfy $\sum_{i=1}^{k} 2^{-\ell(x_i)} = 1$. $\blacksquare$

Next, we find the optimal codeword lengths which satisfy $\sum_{i=1}^{k} 2^{-\ell(x_i)} = 1$. By summing (4.15) over all i, we obtain

$$\mathbb{E}[L] = \frac{\theta + \beta \log 2 - 2a}{3}. \quad (4.19)$$

From (4.15), we obtain

$$-\ell(x_i)+\frac{\beta\log 2}{P_{X_k}(x_i)}2^{-\ell(x_i)}=2\mathbb{E}[L]+2a-\theta, \tag{4.20}$$

for $i\in\{1,2,\dots,k\}$, which yields

$$\frac{\beta(\log 2)^2}{P_{X_k}(x_i)}2^{-\ell(x_i)}e^{\frac{\beta(\log 2)^2}{P_{X_k}(x_i)}2^{-\ell(x_i)}}=\frac{\beta(\log 2)^2}{P_{X_k}(x_i)}2^{\frac{-\theta+2\beta\log 2+2a}{3}}. \tag{4.21}$$

Note that (4.21) is in the form of $xe^x=y$ where the solution for x is equal to $x=W_0(y)$ if $y\geq 0$. Here, $W_0(\cdot)$ denotes the principal branch of the Lambert W function [150]. Since the right hand side of (4.21) is always non-negative, we are only interested in $W_0(\cdot)$ which is denoted as $W(\cdot)$ from now on. We find the unique solution for $\ell(x_i)$ as

$$\ell(x_i)=-\frac{\log\left(\frac{P_{X_k}(x_i)}{\beta(\log 2)^2}W\left(\frac{\beta(\log 2)^2}{P_{X_k}(x_i)}2^{\frac{-\theta+2\beta\log 2+2a}{3}}\right)\right)}{\log 2}, \tag{4.22}$$

for $i\in\{1,2,\dots,k\}$.

In order to find the optimal codeword lengths, we solve (4.22) for a (θ,β) pair that satisfies $p(\theta)=0$ and the Kraft inequality, i.e., $\sum_{i=1}^k 2^{-\ell(x_i)}=1$. Starting from an arbitrary (θ,β) pair, if $p(\theta)>0$ (or $p(\theta)<0$), we increase (or respectively decrease) θ in the next iteration as $p(\theta)$ is a decreasing function of θ. Then, we update β by using (4.19). We repeat this process until $p(\theta)=0$ and $\sum_{i=1}^k 2^{-\ell(x_i)}=1$.

We note that the age-optimal codeword lengths found in this section are for a fixed k. Thus, depending on the selected k, different age performances are achieved at the receiver node. In Section 4.7, we find the age-optimal k values for some given arbitrary pmfs numerically.

Under the highest k selective encoding policy, the receiver node does not receive any update when the remaining $n-k$ realizations occur. However, there may be scenarios in which these remaining realizations are also of interest to the receiver node. In the next section, we focus on this scenario and consider a randomized selection of the remaining $n-k$ realizations so that these realizations are not completely ignored.

4.5 Optimal Codeword Design under Randomized Selective Encoding

The selective encoding scheme discussed so far is a deterministic scheme in which a fixed number of realizations are encoded into codewords and sent to the receiver node when realized. In this section, inspired by [80], we consider a randomized selective encoding scheme where the transmitter encodes the most probable k realizations with probability 1, and encodes the remaining least probable $n-k$ realizations with probability α and thus, neglects them with probability $1-\alpha$. Thus, this randomized selective encoding policy strikes a balance between encoding every single realization and the highest k selective encoding scheme discussed so far.

Theorem 4.1 determines the average age experienced by the receiver node under the randomized highest k selective encoding scheme.

Theorem 4.1 *Under the randomized highest k selective encoding scheme, the av-*

erage age at the receiver node is given by

$$\Delta_\alpha = \frac{\mathbb{E}[L^2] + \frac{2}{q_{k,\alpha}\lambda}\mathbb{E}[L] + \frac{2}{(q_{k,\alpha}\lambda)^2}}{2\left(\mathbb{E}[L] + \frac{1}{q_{k,\alpha}\lambda}\right)} + \mathbb{E}[L], \tag{4.23}$$

where $\mathbb{E}[L] = \sum_{i=1}^{n} P_{X_\alpha}(x_i)\ell(x_i)$*, and* $\mathbb{E}[L^2] = \sum_{i=1}^{n} P_{X_\alpha}(x_i)\ell(x_i)^2$.

The proof of Theorem 4.1 follows similarly to that of Proposition 4.1 by replacing q_k with $q_{k,\alpha}$.

Next, we formulate the age minimization problem for this case as,

$$\begin{aligned} \min_{\{\ell(x_i),\alpha\}} \quad & \frac{\mathbb{E}[L^2] + 2\bar{a}\mathbb{E}[L] + 2\bar{a}^2}{2(\mathbb{E}[L] + \bar{a})} + \mathbb{E}[L] \\ \text{s.t.} \quad & \sum_{i=1}^{n} 2^{-\ell(x_i)} \leq 1 \\ & \ell(x_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, n\}, \end{aligned} \tag{4.24}$$

where the objective function is equal to the average age Δ_α in Theorem 4.1 with $\bar{a} = \frac{1}{\lambda q_{k,\alpha}}$, the first and second constraints follow from the Kraft inequality and the feasibility of the codeword lengths, i.e., each codeword length should be non-negative.

We first solve this problem for a fixed α in this section and determine the optimal α numerically for given arbitrary pmfs in Section 4.7. Following a similar solution technique to that in Section 4.4, we find

$$\ell(x_i) = -\frac{\log\left(\frac{P_{X_\alpha}(x_i)}{\beta(\log 2)^2} W\left(\frac{\beta(\log 2)^2}{P_{X_\alpha}(x_i)} 2^{\frac{-\theta + 2\beta\log 2 + 2\bar{a}}{3}}\right)\right)}{\log 2}, \tag{4.25}$$

for $i \in \{1, 2, \dots, n\}$. To determine the age-optimal codeword lengths $\ell(x_i)$ for $i \in \{1, 2, \dots, n\}$, we then employ the algorithm described in Section 4.4.

In the following section, we consider the case where instead of sending the remaining least probable $n-k$ realizations randomly, the transmitter sends an empty symbol for these updates to further inform the receiver.

4.6 Optimal Codeword Design Under Selective Encoding with an Empty Symbol

In this section, we calculate the average age by considering two different scenarios for the empty symbol. Operationally, the receiver may not reset its age when x_e is received as it is not a regular update packet and the receiver does not know which realization occurred specifically. On the other hand, the receiver may choose to update its age as this empty symbol carries some information, the fact that the current realization is not one of the k encoded realizations, regarding the observed random variable. Thus, in this section, we consider both of these scenarios[6] and find the age-optimal codeword lengths for the set $\mathcal{X}_k'$ with the pmf $\{P_X(x_1), P_X(x_2), \dots, P_X(x_k), P_X(x_e)\}$ in each scenario.

[6]We note that another possible scenario may be to drop the age to an intermediate level between not updating at all and updating fully, considering the partial information conveyed by the empty status update. This case is not considered in this chapter.

4.6.1 When the Empty Symbol Does Not Reset the Age

In this way of operation, the age at the receiver is not updated when the empty status update x_e is received. Thus, sending x_e incurs an additional burden since it does not reset the age but increases the average codeword length of the selected k realizations.

The update cycle is given by (4.6) with

$$W = (M-1)\ell(x_e) + \sum_{\ell=1}^{M} Z_\ell, \tag{4.26}$$

where M is defined in Section 4.4 and denotes the total number of update arrivals until the first update from the set $\mathcal{X}_k$ is observed at the transmitter. In other words, there are $M-1$ deliveries of the empty status update x_e in between two successive deliveries from the encoded set $\mathcal{X}_k$. As discussed earlier, Z is an exponential random variable with rate λ and M is a geometric random variable with parameter q_k. By using the fact that the arrival and service processes are independent, i.e., S and Z are independent, and M is independent of S and Z, in Theorem 4.2, we find the average age when an empty status update does not reset the age.

Theorem 4.2 *When the empty status update x_e does not reset the age, the average age under the highest k selective encoding scheme with an empty symbol at the receiver is given by*

$$\Delta_e = \frac{\mathbb{E}[L^2|X'_k \neq x_e] + 2\mathbb{E}[W]\mathbb{E}[L|X'_k \neq x_e] + \mathbb{E}[W^2]}{2\left(\mathbb{E}[L|X'_k \neq x_e] + \mathbb{E}[W]\right)} + \mathbb{E}[L|X'_k \neq x_e]. \tag{4.27}$$

Proof: We note that the service time of a successful update is equal to its codeword length so that we have

$$\mathbb{E}[S] = \mathbb{E}[L|X'_k \neq x_e] = \sum_{i=1}^{k} P_{X_k}(x_i)\ell(x_i) \tag{4.28}$$

$$\mathbb{E}[S^2] = \mathbb{E}[L^2|X'_k \neq x_e] = \sum_{i=1}^{k} P_{X_k}(x_i)\ell(x_i)^2 \tag{4.29}$$

where $P_{X_k}(x_i)$ is defined in (4.1). By using the independence of M and Z, we find

$$\mathbb{E}[W] = \ell(x_e)\left(\frac{1}{q_k}-1\right)+\frac{1}{\lambda q_k}, \tag{4.30}$$

$$\mathbb{E}[W^2] = \frac{(2-q_k)(1-q_k)}{q_k^2}\ell(x_e)^2+\frac{4(1-q_k)}{\lambda q_k^2}\ell(x_e)+\frac{2}{(\lambda q_k)^2}, \tag{4.31}$$

where we used $\mathbb{E}[M] = \frac{1}{q_k}$, $\mathbb{E}[M^2] = \frac{2-q_k}{q_k^2}$, and Z has exponential distribution with rate λ. Substituting (4.28)-(4.31) in (4.8) yields the result in (4.27). ■

We note that Δ_e in (4.27) depends on $\ell(x_e)$ only through the overall waiting time W as the age does not change when x_e is received. Next, we write the age minimization problem as

$$\begin{aligned}
\min_{\{\ell(x_i),\ell(x_e)\}} \quad & \frac{\mathbb{E}[L^2|X'_k \neq x_e]+2\mathbb{E}[W]\mathbb{E}[L|X'_k \neq x_e]+\mathbb{E}[W^2]}{2\left(\mathbb{E}[L|X'_k \neq x_e]+\mathbb{E}[W]\right)}+\mathbb{E}[L|X'_k \neq x_e] \\
\text{s.t.} \quad & 2^{-\ell(x_e)}+\sum_{i=1}^{k} 2^{-\ell(x_i)} \leq 1 \\
& \ell(x_i) \in \mathbb{R}^+, \quad i \in \{1,\dots,k,e\},
\end{aligned} \tag{4.32}$$

where the objective function is equal to the average age expression Δ_e in (4.27).

We note that problem (4.32) is not convex due to the middle term in the objective function. However, when $\ell(x_e)$ is fixed, it is a convex problem. Thus, we first solve the problem in (4.32) for a fixed $\ell(x_e)$ and then determine the optimal $\ell(x_e)$ numerically in Section 4.7.

Thus, for a fixed $\ell(x_e)$, (4.32) becomes

$$\begin{aligned} \min_{\{\ell(x_i)\}} \quad & \frac{\mathbb{E}[L^2|X_k' \neq x_e] + 2\mathbb{E}[W]\mathbb{E}[L|X_k' \neq x_e] + \mathbb{E}[W^2]}{2\left(\mathbb{E}[L|X_k' \neq x_e] + \mathbb{E}[W]\right)} + \mathbb{E}[L|X_k' \neq x_e] \\ \text{s.t.} \quad & \sum_{i=1}^{k} 2^{-\ell(x_i)} \leq 1 - 2^{-c} \\ & \ell(x_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, k\}, \end{aligned} \tag{4.33}$$

where $\ell(x_e) = c$. Since the empty status update length $\ell(x_e)$ is fixed and given, we write the Kraft inequality by subtracting the portion allocated for $\ell(x_e)$ in the optimization problem in (4.33). Similar to previous sections, we define $p(\theta)$ as

$$\begin{aligned} p(\theta) := \min_{\{\ell(x_i)\}} \quad & \frac{1}{2}\mathbb{E}[L^2|X_k' \neq x_e] + \mathbb{E}[L|X_k' \neq x_e]^2 + (2\hat{a} - \theta)\mathbb{E}[L|X_k' \neq x_e] + \frac{d}{2} - \theta\hat{a} \\ \text{s.t.} \quad & \sum_{i=1}^{k} 2^{-\ell(x_i)} \leq 1 - 2^{-c} \\ & \ell(x_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, k\}, \end{aligned} \tag{4.34}$$

where $\hat{a} = \mathbb{E}[W]$ and $d = \mathbb{E}[W^2]$. For a fixed and given $\ell(x_e)$, the optimization problem in (4.34) is convex. We define the Lagrangian function as

$$\mathcal{L} = \frac{1}{2}\mathbb{E}[L^2|X_k' \neq x_e] + \mathbb{E}[L|X_k' \neq x_e]^2 + (2\hat{a} - \theta)\mathbb{E}[L|X_k' \neq x_e] + \frac{d}{2} - \theta\hat{a}$$

$$+\beta\left(\sum_{i=1}^{k} 2^{-\ell(x_i)} + 2^{-c} - 1\right), \tag{4.35}$$

where $\beta \geq 0$. The KKT conditions are

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \ell(x_i)} =& P_{X_k}(x_i)\ell(x_i) + 2\mathbb{E}[L|X_k' \neq x_e]P_{X_k}(x_i) + (2\hat{a}-\theta)P_{X_k}(x_i) \\ &- \beta(\log 2)2^{-\ell(x_i)} = 0, \end{aligned}\tag{4.36}$$

for all i, and the complementary slackness condition is

$$\beta\left(\sum_{i=1}^{k} 2^{-\ell(x_i)} + 2^{-c} - 1\right) = 0. \tag{4.37}$$

Lemma 4.2 shows that the optimal codeword lengths satisfy $\sum_{i=1}^{k} 2^{-\ell(x_i)} = 1 - 2^{-c}$.

Lemma 4.2 *For the age-optimal real-valued codeword lengths, we must have* $\sum_{i=1}^{k} 2^{-\ell(x_i)} = 1 - 2^{-c}$.

Proof: Assume that the optimal codeword lengths satisfy $\sum_{i=1}^{k} 2^{-\ell(x_i)} < 1 - 2^{-c}$, which implies that $\beta = 0$ due to (4.37). From (4.36), we have

$$P_{X_k}(x_i)\ell(x_i) + 2\left(\sum_{j=1}^{k} P_{X_k}(x_j)\ell(x_j)\right)P_{X_k}(x_i) + (2\hat{a}-\theta)P_{X_k}(x_i) = 0, \quad \forall i. \tag{4.38}$$

By summing (4.38) over all i, we get $\mathbb{E}[L] = \frac{\theta-2\hat{a}}{3}$. Then, we find $\ell(x_i) = \frac{\theta-2\hat{a}}{3}$ for all $i \in \{1,\ldots,k\}$ which makes $p(\theta) = -\frac{\theta^2+2\hat{a}\theta+4\hat{a}^2-3d}{6}$. By using $p(\theta) = 0$, we find

$\theta = -\hat{a} + \sqrt{3(d-\hat{a}^2)}$ which gives $\ell(x_i) = -\hat{a} + \sqrt{\frac{d-\hat{a}^2}{3}}$ for $i \in \{1,\dots,k\}$. One can show that $\frac{\partial\theta}{\partial c} > 0$, i.e., θ, hence age, is an increasing function of c. Thus, in the optimal policy, in order to minimize the average age, c must be equal to zero. However, choosing $c = 0$ leads to $\sum_{i=1}^{k} 2^{-\ell(x_i)} < 1 - 2^{-c} = 0$. Since the sum on the left cannot be negative, we reach a contradiction. Thus, the optimal codeword lengths must satisfy $\sum_{i=1}^{k} 2^{-\ell(x_i)} = 1 - 2^{-c}$. ■

Thus, for the age-optimal codeword lengths, we have $\sum_{i=1}^{k} 2^{-\ell(x_i)} = 1 - 2^{-c}$ and $\beta \geq 0$ from (4.37). By summing (4.36) over all i and using Lemma 4.2 we find

$$\mathbb{E}[L|X_k' \neq x_e] = \frac{\theta + \beta\log 2(1-2^{-c}) - 2\hat{a}}{3}. \tag{4.39}$$

From (4.36), we obtain

$$-\ell(x_i) + \frac{\beta\log 2}{P_{X_k}(x_i)}2^{-\ell(x_i)} = 2\mathbb{E}[L|X_k' \neq x_e] + 2\hat{a} - \theta. \tag{4.40}$$

Thus, we find the unique solution for $\ell(x_i)$ as

$$\ell(x_i) = -\frac{\log\left(\frac{P_{X_k}(x_i)}{\beta(\log 2)^2}W\left(\frac{\beta(\log 2)^2}{P_{X_k}(x_i)}2^{\frac{-\theta+2\beta\log 2(1-2^{-c})+2\hat{a}}{3}}\right)\right)}{\log 2}, \tag{4.41}$$

for $i \in \{1,\dots,k\}$. To determine the age-optimal codeword lengths $\ell(x_i)$ for $i \in \{1,\dots,k\}$, we then employ the algorithm described in Section 4.4.

We note that the average age achieved at the receiver depends on $\ell(x_e)$. In Section 4.7, we provide numerical results where we vary $\ell(x_e)$ over all possible values and choose the one that yields the least average age for given arbitrary pmfs.

4.6.2 When the Empty Symbol Resets the Age

In this subsection, we consider the case where the empty symbol resets the age as it carries *partial* status information as in [84, 87]. In other words, each update which arrives when the transmitter idles is accepted as a successful update.

Theorem 4.3 determines the average age Δ_e when the empty symbol resets the age.

Theorem 4.3 *When the empty status update x_e resets the age, the average age under the highest k selective encoding scheme at the receiver is given by*

$$\Delta_e = \frac{\mathbb{E}[L^2] + 2\frac{1}{\lambda}\mathbb{E}[L] + \frac{2}{\lambda^2}}{2\left(\mathbb{E}[L] + \frac{1}{\lambda}\right)} + \mathbb{E}[L]. \tag{4.42}$$

Proof: Different from the previous sections, the moments for the waiting time are equal to $\mathbb{E}[W] = \frac{1}{\lambda}$ and $\mathbb{E}[W^2] = \frac{2}{\lambda^2}$ as each successful symbol is able to reset the age. Thus, substituting $\mathbb{E}[W]$ and $\mathbb{E}[W^2]$ in (4.8) and noting that $\mathbb{E}[S] = \mathbb{E}[L]$ yields the result. ■

Next, we formulate the age minimization problem as

$$\begin{aligned} \min_{\{\ell(x_i),\ell(x_e)\}} \quad & \frac{\mathbb{E}[L^2] + 2\tilde{a}\mathbb{E}[L] + 2\tilde{a}^2}{2(\mathbb{E}[L] + \tilde{a})} + \mathbb{E}[L] \\ \text{s.t.} \quad & 2^{-\ell(x_e)} + \sum_{i=1}^{k} 2^{-\ell(x_i)} \leq 1 \\ & \ell(x_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, k, e\}, \end{aligned} \tag{4.43}$$

where $\tilde{a} = \frac{1}{\lambda}$. We follow a similar solution technique to that given in Section 4.4 to get

$$\ell(x_i) = -\frac{\log\left(\frac{P_X(x_i)}{\beta(\log 2)^2} W\left(\frac{\beta(\log 2)^2}{P_X(x_i)} 2^{\frac{-\theta+2\beta\log 2+2\tilde{a}}{3}}\right)\right)}{\log 2}, \tag{4.44}$$

for $i \in \{1, \dots, k, e\}$.

The value of k affects $\ell(x_e)$ such that when k is close to n, the probability of the empty symbol becomes small which leads to a longer $\ell(x_e)$, whereas when k is small, the probability of the empty symbol becomes large which results in a shorter $\ell(x_e)$. In Section 4.7, we numerically determine the optimal k selection which achieves the lowest average age for a given arbitrary distribution.

4.7 Numerical Results

In this section, we provide numerical results for the optimal encoding policies that are discussed in Sections 4.4, 4.5, and 4.6. In the first two numerical results, we perform simulations to characterize optimal k values that minimize the average age with the highest k selective encoding scheme in Section 4.4. For these simulations, we use Zipf(n, s) distribution with the following pmf for $n = 100$, $s = 0.4$,

$$P_X(x_i) = \frac{i^{-s}}{\sum_{j=1}^{n} j^{-s}}, \quad 1 \leq i \leq n. \tag{4.45}$$

In Fig. 4.4, we show the effect of sending the most probable k realizations when the update packets arrive at the transmitter node rather infrequently, i.e., the arrival

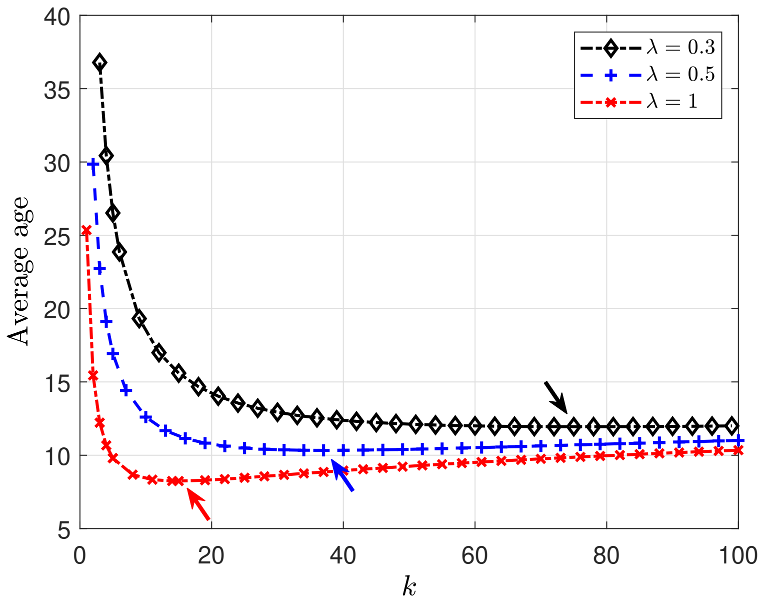


Figure 4.4: The average age values with the age-optimal codeword lengths for $\lambda \in \{0.3, 0.5, 1\}$ for the pmf provided in (4.45) with the parameters $n = 100$, $s = 0.4$. We apply the highest k selective encoding scheme and vary k from 1 to n and indicate k that minimizes the average age for each λ with an arrow.

rate is low. We consider the cases in which the arrival rate is equal to $\lambda = 0.3, 0.5, 1$. For each arrival rate, we plot the average age as a function of $k = 1, 2, \dots, n$. We see that increasing the arrival rate reduces the average age as expected. In this case, optimal k is not equal to 1 since the effective arrival rate is small. In other words, the transmitter node wants to encode more updates as opposed to idly waiting for the next update arrival when the arrivals are rather infrequent. Choosing k close to n is also not optimal as the service times of the status updates with low probabilities are longer which hurts the overall age performance. Indeed, in Fig. 4.4, where update arrival rates are relatively small, it is optimal to choose $k = 76$ for $\lambda = 0.3$, $k = 37$ for $\lambda = 0.5$, and $k = 15$ for $\lambda = 1$.

In Fig. 4.5, we consider a similar setting as in Fig. 4.4 but here update arrival

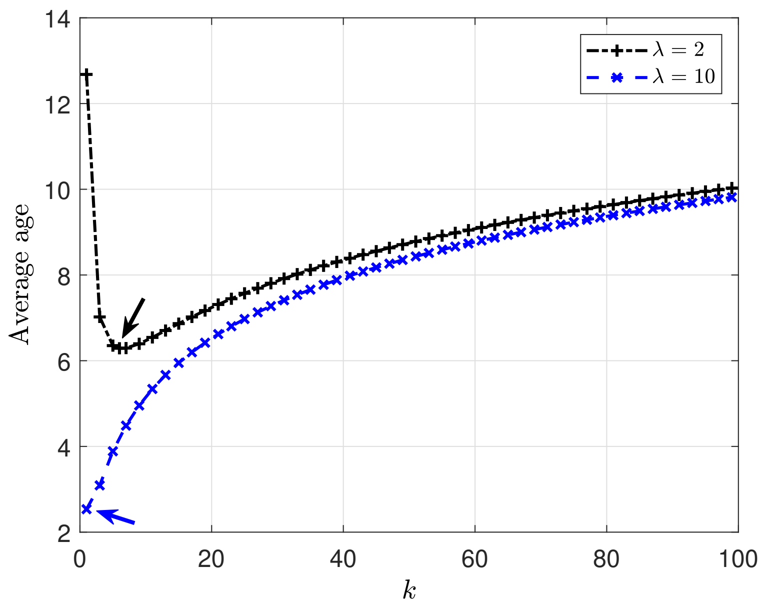


Figure 4.5: The average age values with the age-optimal codeword lengths for $\lambda \in \{2, 10\}$ for the pmf provided in (4.45) with the parameters $n = 100$, $s = 0.4$. We apply the highest k selective encoding scheme and vary k from 1 to n and observe that choosing $k = 1$ under the relatively high arrival rates ($\lambda = 10$) minimizes the average age.

rates are larger which means that updates arrive more frequently at the transmitter node. We observe that when $\lambda = 2$, the optimal k is still not equal to 1 (it is equal to 6 in Fig. 4.5) as the updates are not frequent enough. However, once updates become more available to the transmitter node, i.e., the case with $\lambda = 10$ in Fig. 4.5, we observe that the transmitter node chooses to only encode the realization with the highest probability, i.e., $k = 1$, and wait for the next update arrival instead of encoding more and more realizations which increases the overall codeword lengths thereby increasing the transmission times. We also observe that the average age decreases as the update arrival rate increases as in Fig. 4.4.

We note that when the arrival rate is high as in Fig. 4.5 when $\lambda = 10$, we observe that the age is an increasing function of k since under this arrival profile

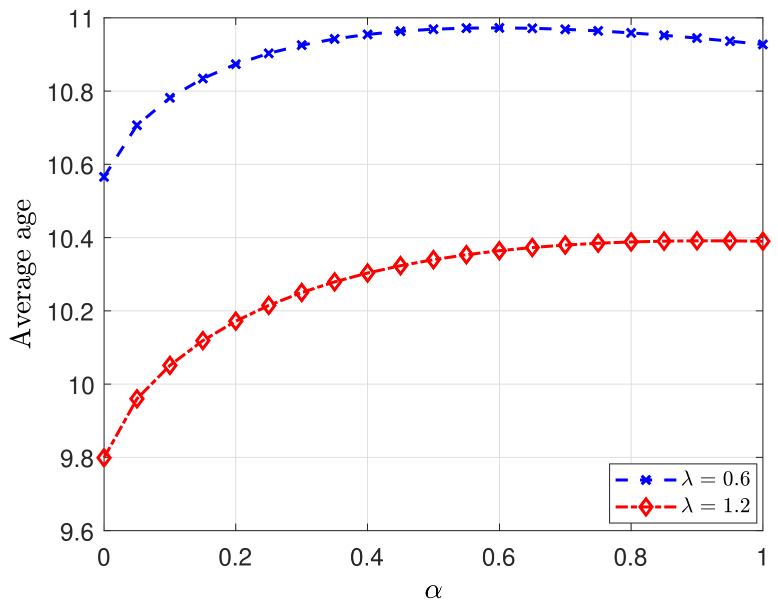


Figure 4.6: The average age values with the age-optimal codeword lengths for different α values with the pmf provided in (4.45) with $n = 100$, $s = 0.2$ for $k = 70$ and $\lambda = 0.6, 1.2$ when randomized highest k selective encoding is implemented.

codeword lengths dominate the performance which in turn increase as k increases. On the other hand, when the arrival rate is low as in $\lambda = 2$ in Fig. 4.5 and $\lambda = 0.3, 0.5, 1$ in Fig. 4.4, we observe that initially the age is a decreasing function of k as the waiting time in between two successive encoded updates dominates the performance. However, when we continue to increase k, we observe that both of these opposing trends are in play and the age starts to increase with k.

For the third numerical result shown in Fig. 4.6, we simulate the randomized highest k selective encoding policy described in Section 4.5 with Zipf distribution in (4.45) with parameters $n = 100$, $s = 0.2$. In Fig. 4.6, we observe two different trends depending on the update arrival frequency at the source node, even though in either case, randomization results in a higher age at the receiver node than selective encoding, i.e., $\alpha = 0$ case. When the arrival rate is high, $\lambda = 1.2$ in Fig. 4.6, we observe that age monotonically increases with α as randomization increases average codeword length, i.e., service times. Although increasing α results in a higher age at the receiver node, previously discarded $n-k$ realizations can be received under this randomized selective encoding policy. Interestingly, when the arrival rate is smaller, $\lambda = 0.6$ in Fig. 4.6, we observe that age initially increases with α and then starts to decrease because of the decreasing waiting times as opposed to increasing codeword lengths such that when α is larger than 0.3, it is better to select $\alpha = 1$, i.e., encoding every realization. That is, when α grows beyond 0.3, encoding and sending every single realization yields a lower average age.

In the fourth and fifth numerical results, we find the optimal real-valued codeword lengths and k values that minimize the average age Δ_e with the highest k selective encoding scheme with an empty symbol, as discussed in Section 4.6. For these numerical results, we use the following pmf

$$P_X(x_i) = \begin{cases} 2^{-i}, & i = 1, \dots, n-1 \\ 2^{-n+1}, & i = n. \end{cases} \tag{4.46}$$

In the fourth numerical result, we consider the pmf in (4.46) for $n = 10$ and take $\lambda = 5$. We find the optimal codeword length of the empty symbol, $\ell(x_e)$, when the empty symbol does not reset the age (see Fig. 4.7). We observe that when k is small, the probability of sending the empty symbol becomes large so that a shorter codeword is preferable for x_e. For example, we observe in Fig. 4.7 that choosing $\ell(x_e) = 2$ when $k = 2$ and $\ell(x_e) = 3$ when $k = 4$ is optimal. Similarly, when k is

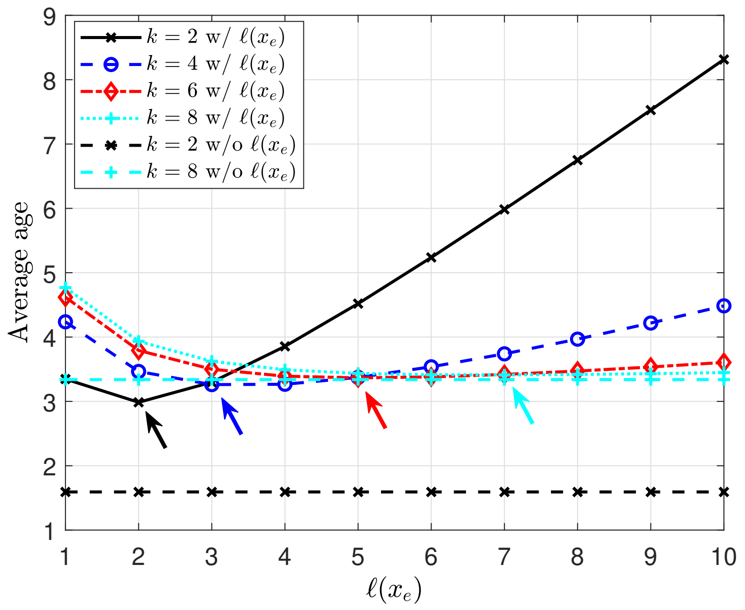


Figure 4.7: Average age with the age-optimal codeword lengths with respect to $\ell(x_e)$ with the pmf in (4.46) for $n = 10$ and $\lambda = 5$ when the empty symbol does not reset the age. Arrows indicate the age-optimal $\ell(x_e)$ values. We also provide the optimal age without sending the empty symbol for $k = 2$ and $k = 8$.

larger, a longer codeword is desirable for x_e. We observe in Fig. 4.7 that choosing $\ell(x_e) = 5$ when $k = 6$ and $\ell(x_e) = 7$ when $k = 8$ is optimal. Further, we note in Fig. 4.7 that the average age increases when we send the empty symbol in the case of the remaining $n - k$ realizations as the empty symbol increases the total waiting time for the next successful arrival as well as the codeword lengths for the encoded k realizations. For smaller k values, i.e., when $k = 2$, this effect is significant as the empty symbol has a large probability whereas when k is larger, i.e., when $k = 8$, sending an empty status update increases the age slightly (especially when $\ell(x_e)$ is high) as the empty symbol has a small probability.

In the fifth numerical result shown in Fig. 4.8, we consider the case when the empty symbol x_e resets the age. We observe that the minimum age is achieved

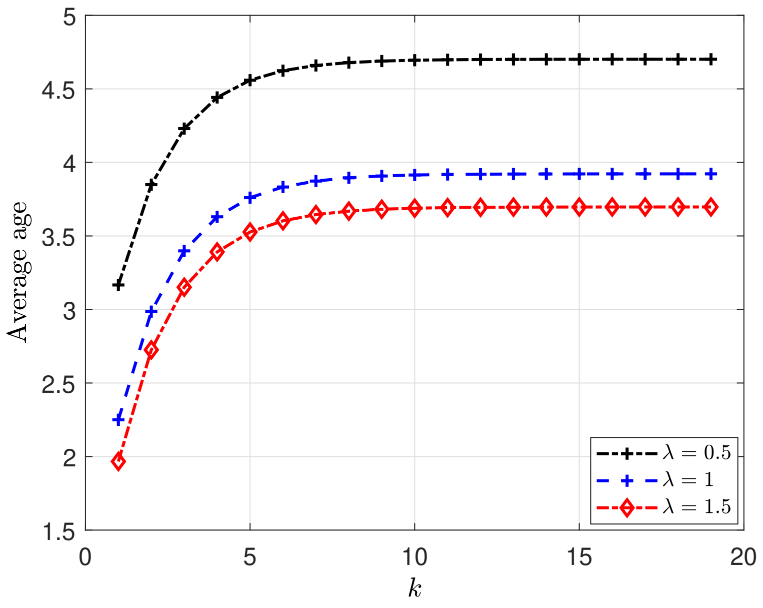


Figure 4.8: Average age with the age-optimal codeword lengths for varying k with the pmf in (4.46) for $n = 20$ and $\lambda = 0.5, 1, 1.5$ when the empty symbol resets the age.

when $k = 1$, i.e., only the most probable realization is encoded. This is because the overall waiting time is independent of k and larger k values result in larger codewords which in turn increases transmission times. Thus, in this case, only the most probable realization is received separately since all others are embedded into the empty symbol. We note that this selection results in significant information loss at the receiver which is not captured by the age metric alone. This problem can be addressed by introducing a distortion constraint which measures the information loss together with the age metric which measures freshness [87].

In the sixth numerical result shown in Fig. 4.9, we compare the performance of the age-optimal code that we developed in Section 4.4 with well-known codes that minimize average codeword length. For this purpose, we choose Huffman code[7]

[7]We acknowledge the feedback from one of the anonymous Reviewers who suggested that we compare our age-optimal code with Huffman codes.

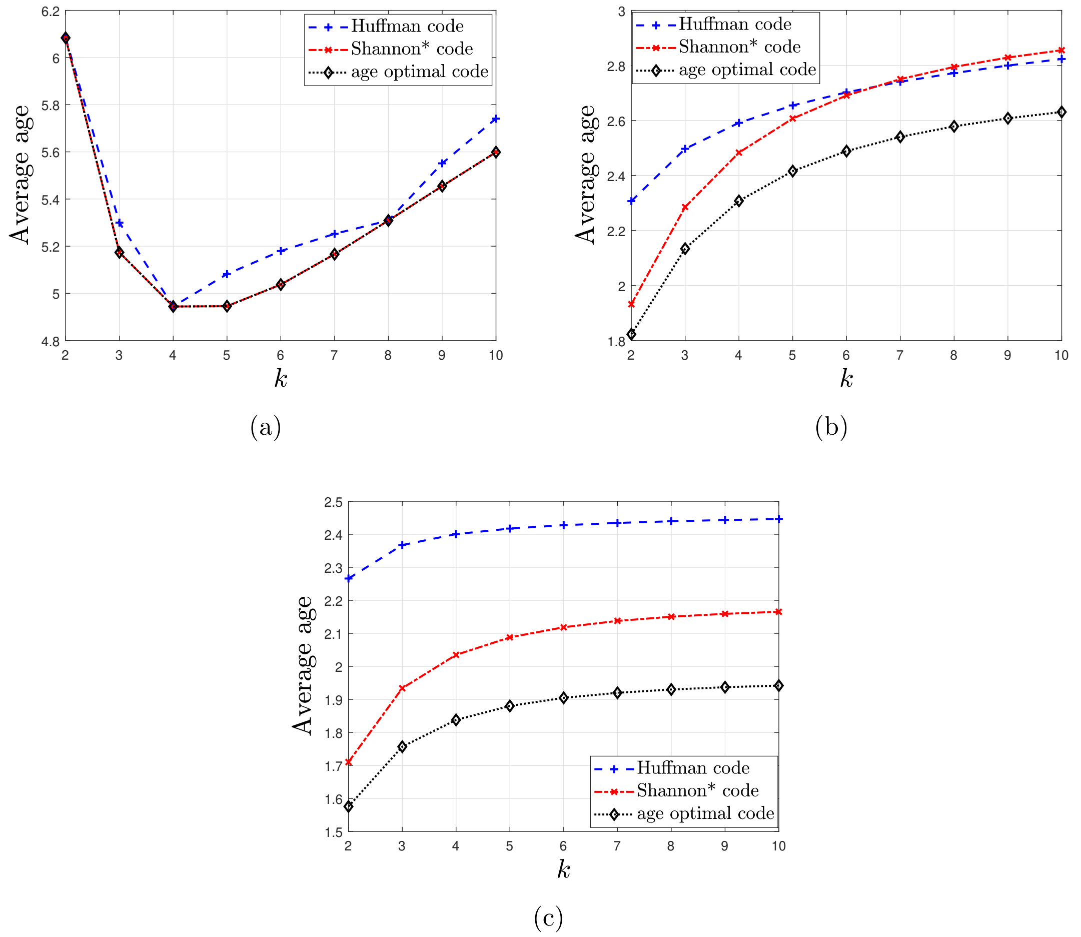


Figure 4.9: The average age under Huffman code, Shannon* code and the age-optimal code for $\lambda = 1$ and the pmf in (4.45) with the parameters $n = 10$, (a) $s = 0$, (b) $s = 3$ and (c) $s = 4$. We vary k from 2 to n.

which takes integer-valued codeword lengths and Shannon* code[8] which takes real-valued codeword lengths. We use the pmf in (4.45) with $n = 10$ and $s = 0, 3, 4$ for $\lambda = 1$.

We note that when $s = 0$, the distribution in (4.45) becomes a uniform distribution. We see in Fig. 4.9(a) that for the uniform distribution, the age-optimal

[8]We note that codeword lengths of a Shannon code are equal to $\lceil -\log_2(P_{X_k}(x_i)) \rceil$ which take integer values [146]. However, in this chapter, we neglect the ceiling operator and consider real-valued codeword lengths as $-\log_2(P_{X_k}(x_i))$ which we denote as a Shannon* code.

real-valued codeword lengths are equivalent to Shannon* code. This result has been observed in [80] as well. When k is equal to a power of 2 such as $k = 2, 4, 8$ in Fig. 4.9(a), Huffman code becomes the same as Shannon* code as the codeword lengths of Shannon* code, i.e., $-\log_2(P_{X_k}(x_i))$, take integer values. For the remaining k values Huffman code performs worse than Shannon* code and the age-optimal code. When $s = 3$, we see in Fig. 4.9(b) that the age-optimal code achieves a smaller age than Huffman and Shannon* codes. When $k < 7$, we see in Fig. 4.9(b) that Shannon* code achieves a lower age than Huffman code whereas when $k \geq 7$, Huffman code achieves a lower age than Shannon code. When $s = 4$, we see in Fig. 4.9(c) that the age-optimal code achieves the lowest age whereas Huffman code performs the worst.

Thus, we observe that when the distribution is close to a uniform distribution, i.e., when s is small, Huffman and Shannon* codes perform similar to the age-optimal code (when the distribution is equal to uniform distribution, we see that Shannon* code is equivalent to the age-optimal code). However, when the distribution is more polarized, i.e., when s is high, we see that the age-optimal code performs significantly better than Shannon* and Huffman codes.

4.8 On The Optimality of the Highest k Selective Encoding

So far, we have considered only the case where the most probable k realizations are encoded and sent through the channel. Based on this selection, we found the average age and determined the age-optimal k and codeword lengths. We observed that this highest k selective encoding policy results in a lower average age than encoding every realization. However, we note that there are $\binom{n}{k}$ selections for encoding and in this section, we discuss the optimality of the highest k selective encoding among all these different selections. We see that the average age expression in Proposition 4.1 depends on the pmf of X which affects the optimal codeword lengths, and the effective arrival rate. In this section, we denote the effective arrival rate as λ_e given by $\lambda_e = \lambda \sum_{x \in \mathcal{X}_s} P_X(x)$ where $\mathcal{X}_s$ is the set of arbitrarily selected k updates for encoding. Here, by choosing a different set of k realizations to encode and send, instead of the most probable k realizations, we change the effective arrival rate and codeword lengths which in turn yields a different age performance.

When the arrival rate is relatively low, we see in Fig. 4.4 that the average age is dominated mainly by the effective arrival rate. Thus, choosing the realizations with the highest probabilities may be desirable as this selection achieves the highest possible effective arrival rate. However, when the arrival rate is relatively high, the average age is mainly determined by the moments of the codeword lengths.

In Table 4.1, we find the age-optimal update selections for given pmfs and arrival rates for $k = 5$. We use the in (4.46) with $n = 10$ and Zipf distribution in (4.45) with parameters $n = 10$, $s = 0.2$. In both pmfs, the updates are in decreasing order with respect to their probabilities, i.e., $P_X(x_i) \geq P_X(x_j)$ if $i \leq j$. When the arrival rate is relatively small, i.e., $\lambda = 0.1$ for the first pmf and $\lambda = 0.5$ for the second pmf, we observe that choosing the realizations with the highest probabilities for encoding is optimal as this selection increases the effective arrival rate the most which is the dominating factor for the age performance when

pmf	λ	optimal selection	λ_e	optimal age
The pmf in (4.46) for $n = 10$	0.1	$\{1, 2, 3, 4, 5\}$	0.0969	12.292
	0.5	$\{1, 2, 8, 9, 10\}$	0.3789	3.867
	1	$\{1, 7, 8, 9, 10\}$	0.5156	2.4229
Zipf($n = 10$, $s = 0.2$)	0.5	$\{1, 2, 3, 4, 5\}$	0.3898	5.154
	1	$\{1, 2, 8, 9, 10\}$	0.6269	3.929
	2	$\{1, 7, 8, 9, 10\}$	1.01	3.304

Table 4.1: Age-optimal update selection for fixed $k = 5$ with different arrival rates, λ.

the arrivals are infrequent at the source node. That is, the optimal selection is $\{1, 2, 3, 4, 5\}$ when $\lambda = 0.1$ for the first pmf and when $\lambda = 0.5$ for the second pmf. However, when the arrival rate is high, the optimal policy is to encode the realization with the highest probability and $k-1$ realizations with the lowest probabilities such that the optimal set is $\{1, 7, 8, 9, 10\}$ as this selection helps to keep the moments of codewords lengths at appropriate levels which are the dominating factors for the age performance when the arrivals are frequent at the source node. We see that this selection is optimal when $\lambda = 1$ for the first pmf and when $\lambda = 2$ for the second pmf. From these, we observe that the optimal update selection strategy is to keep the effective arrival rate as high as possible while maintaining the moments of the codeword lengths at the desired levels. We see this structure when $\lambda = 0.5$ for the first pmf and $\lambda = 1$ for the second pmf where the optimal selection is to choose the most probable two and the least probable three realizations, i.e., the optimal selection is $\{1, 2, 8, 9, 10\}$.

Thus, even though the highest k selective encoding policy improves the age performance as shown in Section 4.7, this selection may not necessarily be optimal for a given pmf and arrival rate among all other possible selections. In fact, in

Table 4.1 we observe that, the highest k selection is optimal when the arrival rate is low. When the arrival rate is high, however, a different k selection should be implemented to get a better age performance as shown in Table 4.1. The theoretical analysis for the optimality of the highest k selective encoding remains as a future work. Further, in some cases the realizations with lower probabilities may carry important information that cannot be ignored. In these scenarios, an importance metric can be assigned to each realization and the encoded k realizations can be selected considering both the importance metric and the realization probabilities. We leave this problem for future work.

4.9 Conclusion

In this chapter, we considered a status updating system in which an information source generates independent and identically distributed update packets based on an observed random variable X which takes n values based on a known pmf. We studied three different encoding schemes for the transmitter node to send the realizations to the receiver node. In all these schemes, the most probable k update realizations are always encoded. For the remaining less probable $n-k$ realizations, we considered the case in which these realizations are completely discarded, i.e., the highest k selective encoding scheme. Next, we considered the case in which the remaining previously discarded $n-k$ realizations are encoded into codewords randomly to further inform the receiver, i.e., randomized selective encoding scheme. Lastly, we examined the case where the remaining less probable realizations are mapped into an empty symbol to partially inform the receiver node, i.e., highest k selective encoding scheme with an empty symbol. We derived the average age for all these encoding schemes and determined the age-optimal codeword lengths. Through numerical results we showed that the proposed selective encoding scheme achieves a lower average age than encoding all the realizations, and determined the age-optimal k values for arbitrary pmfs. We investigated the optimality of the highest k selective encoding and showed through simulations that it is optimal when the arrival rate is low.

CHAPTER 5

Lossy Source Coding with Partial Updates: Losing Information for Freshness

5.1 Introduction

In this chapter, we consider a communication system, shown in Fig. 5.1, where a source generates updates as soon as requested by a transmitter. After an update is generated by the source, the transmitter further processes it to generate a partial update, and assigns a codeword to it using a binary alphabet. The transmitter sends this codeword to the receiver through a noiseless channel. Thus, the transmission time, i.e., the service time, for a partial update is equal to its codeword length. The average service time is equal to the expected codeword length, but the average age depends on both the first and second moments of the codeword lengths. Our goal is to optimize the partial update generation process, and the following codebook design, to minimize the age while maintaining a desired level of fidelity for the partial updates.

References that are most closely related to our work are [77–79, 81, 82, 84]. Reference [77] considers age-optimal block code design and relates age to error ex-

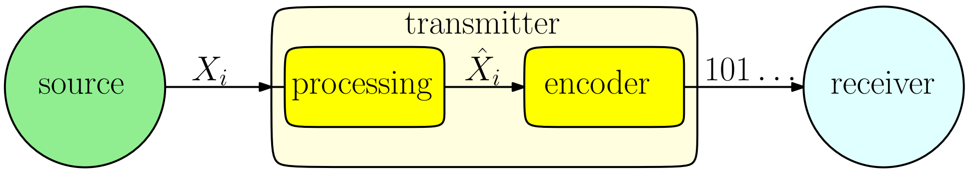


Figure 5.1: An information updating system which consists of a source, a transmitter and a receiver.

ponents. Reference [78] considers the problem of assigning codewords to updates to minimize the average peak age. Reference [79] considers the problem of assigning real-valued codeword lengths to updates to minimize the average age, and shows that Shannon codes can be used with a modified pmf to achieve asymptotically optimal performance. Reference [81] considers the problem of *selectively* encoding a given number of most probable updates while dropping the remaining least probable updates, and shows that this may yield better age performance than encoding all of the realizations. Reference [82] considers the problem of sending an empty status update in the selective encoding scheme of [81] to partially inform the receiver about the dropped updates, i.e., once an empty status update is received, the receiver knows that one of the dropped updates is realized but does not know which one specifically. Finally, reference [84] introduces the concept of partial updates where both the information content and the transmission time are reduced compared to the original updates.

In this chapter, we consider the problem of generating partial updates from the original updates, in a way to minimize the average age, while keeping the *information content* of the partial updates at a desired level. In order to quantify the information similarity between the original updates X and the partial updates $\hat{X}$, we use the mutual information between X and $\hat{X}$. Since $I(X;\hat{X}) = H(\hat{X}) - H(\hat{X}|X) = H(\hat{X})$ as the partial updates $\hat{X}$ are functions of the original updates X, we need to minimize the age while keeping the entropy of the partial updates, $H(\hat{X})$, at a desired level. Thus, our problem reduces to finding a pmf for the partial updates $\hat{X}$ that can be generated from the given pmf of the original updates such that it yields the desired entropy, and the corresponding codeword lengths generated from the pmf of partial updates that minimize the age. This problem is NP-hard as we need to search over all partitioning of the realizations of the original updates to obtain the partial updates. We relax the problem to optimize over all pmfs for partial updates. While the resulting problem is non-convex, it is individually convex with respect to the pmf given the codewords lengths and vice versa. Thus, we develop an alternating minimization based iterative algorithm that optimizes one set of parameters (e.g., pmf) given the other set of parameters (e.g., codeword lengths). We investigate the tradeoff between the average age and the conveyed information content via numerical results.

5.2 System Model and Problem Formulation

We consider a communication system where a source generates independent and identically distributed status updates from a set $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ with a pmf $P_X(x_i) = \{p_1, p_2, \dots, p_n\}$ which is known. Without loss of generality, we assume that $p_i \geq p_j$ if $i < j$, i.e., the elements of the set $\mathcal{X}$ are sorted in decreasing order with respect to their probabilities. The transmitter requests an update from the source once the transmission of the previous update is completed. Thus, the source follows a generate at will model, and the transmitter follows a zero-wait model.

After an update is received by the transmitter, it further processes the update by using a function $g(X)$ to generate a partial update. The function $g(X)$ maps each update $x_i \in \mathcal{X}$ to the set of partial updates $\hat{\mathcal{X}}$, i.e., $g : \mathcal{X} \to \hat{\mathcal{X}}$, where the cardinality of $\hat{\mathcal{X}}$ is k, and $1 \leq k \leq n$. When $k < n$, the transmitter maps some of the original updates from the set $\mathcal{X}$ to one partial update from the set $\hat{\mathcal{X}}$. When $k = n$, the transmitter sends the updates generated from the source without processing, i.e., $g(x) = x$ for all $x \in \mathcal{X}$. We write the pmf of the partial updates as $P_{\hat{X}}(\hat{x}_i) = \{\hat{p}_i | \hat{p}_i = \sum_{i \in S_i} p_i, S_i = \{j | g(x_j) = \hat{x}_i, j = 1, \dots, n\}, i = 1, \dots, k\}$.

For example, let us consider a source which generates an update from $\mathcal{X} = \{a, b, c, d\}$ with pmf $\{0.5, 0.25, 0.125, 0.125\}$. The transmitter processes the updates to generate $k = 3$ different partial updates. Let the set for the partial updates $\hat{\mathcal{X}}$ be $\{\{a\}, \{b\}, \{c, d\}\}$ with corresponding pmf $\{0.5, 0.25, 0.25\}$. When update a or b is realized at the source, the receiver fully knows the realized update once the corresponding partial update is received. However, when update c or d is realized at the source, the partial update $\{c, d\}$ is transmitted. Once it is received, the receiver has the partial information about the update generated at the source, i.e., it knows that c or d is realized but does not know which one is realized specifically.

After generating the partial updates, the transmitter assigns codewords $c(\hat{x}_i)$ with lengths $\ell(\hat{x}_i)$ to each partial update $\hat{x}_i \in \hat{\mathcal{X}}$ by using a binary alphabet. Let the first and second moments of the codeword lengths be $\mathbb{E}[L]$ and $\mathbb{E}[L^2]$ where $\mathbb{E}[L] = \sum_{i=1}^{k} P_{\hat{X}}(\hat{x}_i)\ell(\hat{x}_i)$ and $\mathbb{E}[L^2] = \sum_{i=1}^{k} P_{\hat{X}}(\hat{x}_i)\ell(\hat{x}_i)^2$. We assume that the channel between the transmitter and the receiver is noiseless. Thus, if update $\hat{x}_i$ is transmitted, it takes $\ell(\hat{x}_i)$ units of time to deliver this partial update to the receiver, i.e., $\ell(\hat{x}_i)$ is the system service time for partial update i.

In order to quantify the information retained by the partial updates, we use mutual information $I(X;\hat{X}) = H(\hat{X}) - H(\hat{X}|X)$. We consider AoI to quantify the freshness of the information at the receiver. Let $a(t)$ be the instantaneous age at time t, with $a(0) = 0$. When there is no update, the age at the receiver increases linearly over time. When an update is received, the age at the receiver reduces to the age of the latest received update. Let Δ_T be the average AoI in the time interval $[0, T]$, which is given as

$$\Delta_T = \frac{1}{T}\int_0^T a(t)dt, \tag{5.1}$$

and let Δ be the long term average AoI, i.e., $\Delta = \lim\limits_{T\to\infty} \Delta_T$. The age function evolves as in Fig. 5.2. Given that there are m updates until time T, we write the average AoI, Δ_T, as

$$\Delta_T = \frac{1}{T}\left(\frac{1}{2}\sum_{i=1}^{m} s_i^2 + \sum_{i=1}^{m-1} s_i s_{i+1} + \frac{r^2}{2} + \frac{s_N r}{2}\right), \tag{5.2}$$

where $r = T - \sum_{i=1}^{m} s_i$ and s_i is the service time for the ith realized update. By using similar arguments as in [2], we calculate the long term average AoI, Δ, as

$$\Delta = \lim_{T\to\infty} \Delta_T = \frac{\mathbb{E}[S^2]}{2\mathbb{E}[S]} + \mathbb{E}[S], \tag{5.3}$$

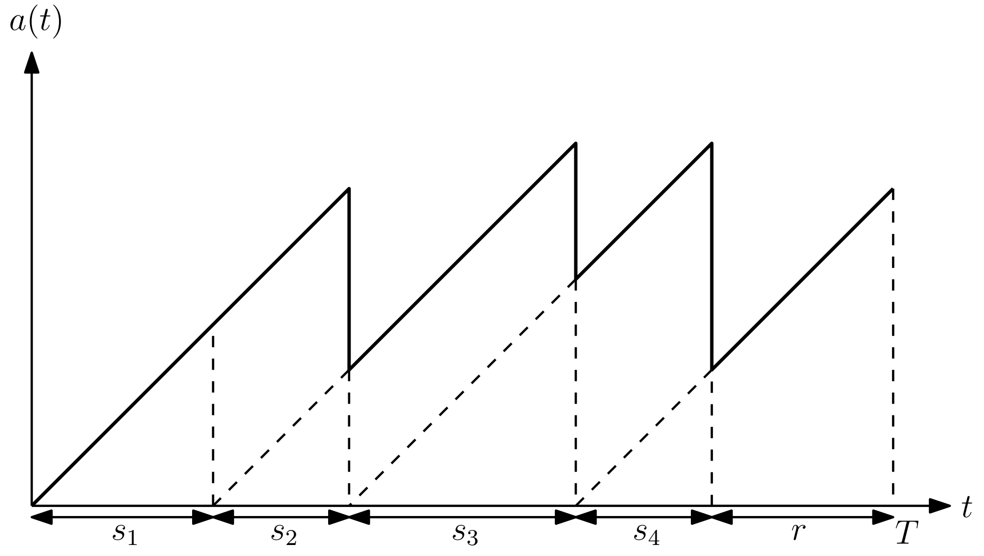


Figure 5.2: Sample age evolution at the receiver.

where we use $\lim\limits_{T\to\infty} \frac{m}{T} = \frac{1}{\mathbb{E}[S]}$, $\lim\limits_{m\to\infty} \frac{\sum_{i=1}^{m} s_i^2}{2m} = \frac{\mathbb{E}[S^2]}{2}$ and $\lim\limits_{m\to\infty} \frac{\sum_{i=1}^{m-1} s_i s_{i+1}}{m} = \mathbb{E}[S]^2$. We note that the moments of the service times are equal to the moments of the codeword lengths, as service times here are codeword lengths, and thus, we have $\mathbb{E}[S] = \mathbb{E}[L]$ and $\mathbb{E}[S^2] = \mathbb{E}[L^2]$.

In this chapter, for a given k, our aim is to find the partial updates and the corresponding codeword lengths to minimize the average age while satisfying the constraints on the mutual information between the original and the partial updates, i.e., $I(X;\hat{X}) = \beta$ where β is the desired level of mutual information, and the feasibility of the codeword lengths expressed by Kraft's inequality, i.e., $\sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} \leq 1$. Therefore, we write the optimization problem as

$$\begin{aligned} \min_{\{\hat{p}_i, \ell(\hat{x}_i)\}} \quad & \Delta \\ \text{s.t.} \quad & I(X;\hat{X}) = \beta \end{aligned}$$

$$
\begin{aligned}
& \sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} \leq 1 \\
& \ell(\hat{x}_i) \in \mathbb{Z}^+, \quad i \in \{1, \dots, k\}.
\end{aligned} \tag{5.4}
$$

We study the optimization problem in (5.4) in the next section.

5.3 The Optimal Solution

In this section, we solve a relaxed version of the problem in (5.4) where we allow codeword lengths to be real-valued. We need to find the age-optimal pmf for the partial updates and the corresponding codeword lengths. We note that the constraint on the mutual information in (5.4), $I(X;\hat{X}) = \beta$, is equivalent to $H(\hat{X}) = \beta$ since $I(X;\hat{X}) = H(\hat{X}) - H(\hat{X}|X)$ and $H(\hat{X}|X) = 0$. Thus, using the age expression in (5.3), the relaxed optimization problem becomes,

$$
\begin{aligned}
\min_{\{\hat{p}_i, \ell(\hat{x}_i)\}} \quad & \frac{\mathbb{E}[L^2]}{2\mathbb{E}[L]} + \mathbb{E}[L] \\
\text{s.t.} \quad & H(\hat{X}) = \beta \\
& \sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} \leq 1 \\
& \ell(\hat{x}_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, k\}.
\end{aligned} \tag{5.5}
$$

Note that in order to solve the optimization problem in (5.5), we need to find a partition of the original updates that produces the age-optimal pmf for the partial updates and the corresponding optimal real-valued codeword lengths. For a given pmf, $\hat{p}_i$, we obtain the age-optimal real-valued codeword lengths in Section 5.3.1.[1] However, finding the optimal partition of the original updates is similar to a bin packing problem which is a well-known combinatorial NP-hard problem [151]. Thus, the problem in (5.5) is NP-hard and the optimal solution can be found by searching over all possible partitions.

In order to progress on the problem analytically, we relax the pmf constraint, and allow all possible pmfs for the partial updates. Note that originally the pmfs are limited only to the pmfs that can be generated from the partitions of n original updates to k partial updates; here, we allow all valid pmfs. Thus, we write the further relaxed problem as

$$\begin{aligned} \min_{\{\hat{p}_i, \ell(\hat{x}_i)\}} \quad & \frac{\mathbb{E}[L^2]}{2\mathbb{E}[L]} + \mathbb{E}[L] \\ \text{s.t.} \quad & H(\hat{X}) = \beta \\ & \sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} \leq 1 \\ & \sum_{i=1}^{k} \hat{p}_i = 1 \\ & \hat{p}_i \geq 0, \quad \ell(\hat{x}_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, k\}. \end{aligned} \tag{5.6}$$

Next, we define $p(\lambda)$ for the problem in (5.6) as

$$p(\lambda) = \min_{\{\hat{p}_i, \ell(\hat{x}_i)\}} \quad \frac{\mathbb{E}[L^2]}{2} + \mathbb{E}[L]^2 - \lambda \mathbb{E}[L]$$

[1]We note that finding the age-optimal codeword lengths has been considered in [79]. Even though this is not our main contribution, we solve this problem here for completeness, and present an alternative technique to [79].

$$
\begin{aligned}
\text{s.t.}\quad & H(\hat{X}) = \beta \\
& \sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} \leq 1 \\
& \sum_{i=1}^{k} \hat{p}_i = 1 \\
& \hat{p}_i \geq 0, \quad \ell(\hat{x}_i) \in \mathbb{R}^+, \quad i \in \{1, \dots, k\}.
\end{aligned} \tag{5.7}
$$

This approach was introduced in [148] and has been used in [31, 46]. One can show that $p(\lambda)$ is a decreasing function of λ and the optimal solution is obtained when $p(\lambda) = 0$. The optimal age for the problem in (5.7) is equal to λ, i.e., $\Delta^* = \lambda$.

We introduce the Lagrangian function [149] for (5.7) as

$$
\begin{aligned}
\mathcal{L} =& \frac{\mathbb{E}[L^2]}{2} + \mathbb{E}[L]^2 - \lambda \mathbb{E}[L] + \theta \left(\sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} - 1 \right) + \gamma \left(\sum_{i=1}^{k} \hat{p}_i \log \hat{p}_i + \beta \right) \\
& + \sigma \left(\sum_{i=1}^{k} \hat{p}_i - 1 \right) - \sum_{i=1}^{k} \nu_i \hat{p}_i - \sum_{i=1}^{k} \mu_i \ell(\hat{x}_i),
\end{aligned} \tag{5.8}
$$

where $\theta \geq 0$, $\nu_i \geq 0$, $\mu_i \geq 0$, and γ and σ can be anything. Next, we write the KKT conditions as

$$
\frac{\partial \mathcal{L}}{\partial \ell(\hat{x}_i)} = \hat{p}_i \ell(\hat{x}_i) + 2\hat{p}_i \mathbb{E}[L] - \lambda p_i - \theta (\log 2) 2^{-\ell(\hat{x}_i)} - \mu_i = 0, \tag{5.9}
$$

$$
\frac{\partial \mathcal{L}}{\partial \hat{p}_i} = \frac{1}{2} \ell(\hat{x}_i)^2 + 2\ell(\hat{x}_i) \mathbb{E}[L] - \lambda \ell(\hat{x}_i) + \gamma \left(\log \hat{p}_i + \frac{1}{\log 2} \right) + \sigma - \nu_i = 0, \tag{5.10}
$$

for all i. The complementary slackness conditions are

$$\theta\left(\sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} - 1\right) = 0, \tag{5.11}$$

$$\gamma\left(\sum_{i=1}^{k} \hat{p}_i \log \hat{p}_i + \beta\right) = 0, \tag{5.12}$$

$$\sigma\left(\sum_{i=1}^{k} \hat{p}_i - 1\right) = 0, \tag{5.13}$$

$$\nu_i \hat{p}_i = 0, \tag{5.14}$$

$$\mu_i \ell(\hat{x}_i) = 0. \tag{5.15}$$

For the partial updates with $\hat{p}_i = 0$, we assign $\ell(\hat{x}_i) = \infty$ as these updates never realize and we reserve the Kraft inequality for the updates with strictly positive probabilities. For the partial updates with $\hat{p}_i > 0$, we have $\nu_i = 0$ from (5.14). As entropy constraint $\beta > 0$, we need at least two updates with $\hat{p}_i > 0$. For each $\hat{p}_i > 0$, we have $\ell(\hat{x}_i) > 0$. Otherwise, if we have $\ell(\hat{x}_i) = 0$ for an update with $\hat{p}_i > 0$, other codeword lengths with non-zero probability have infinite lengths due to the Kraft inequality which clearly cannot be the optimal solution. Thus, we have $\ell(\hat{x}_i) > 0$ which implies $\mu_i = 0$.

We note that the optimization problem in (5.7) is not convex as $\hat{p}_i$s and $\ell(\hat{x}_i)$s appear as multiplicative terms. However, for a given proper $\hat{p}_i$, the problem in (5.7) is convex with respect to the codeword lengths $\ell(\hat{x}_i)$. Similarly, for a given $\ell(\hat{x}_i)$, the problem in (5.7) is convex with respect to $\hat{p}_i$. We apply the alternating minimization method (see e.g., [152–155]) to find $(\hat{p}_i, \ell(\hat{x}_i))$ such that (5.9) and (5.10) are satisfied for all i. Starting with an initial pmf $\hat{p}_i$, we find the optimum real-valued codeword lengths $\ell(\hat{x}_i)$ for the initial pmf $\hat{p}_i$. Then, for given codeword lengths $\ell(\hat{x}_i)$, we find the pmf that is proper, i.e., $\sum_{i=1}^k \hat{p}_i = 1$ and satisfies the entropy condition, i.e., $\sum_{i=1}^k \hat{p}_i \log \hat{p}_i + \beta = 0$. We repeat these steps until the KKT conditions in (5.9) and (5.10) are satisfied.

In Section 5.3.1, we solve for optimum $\ell(\hat{x}_i)$ for given $\hat{p}_i$; in Section 5.3.2, we solve for optimum $\hat{p}_i$ for given $\ell(\hat{x}_i)$; and in Section 5.3.3, we give the iterative algorithm.

5.3.1 Age-Optimal Codeword Lengths for a Given PMF

In this section, we find the age-optimal real-valued codeword lengths for a given pmf which satisfy (5.9). First, we show that for a given pmf, the age-optimal codeword lengths should satisfy the Kraft inequality with equality.

Lemma 5.1 *For the age-optimal real-valued codeword lengths, we have* $\sum_{i=1}^k 2^{-\ell(\hat{x}_i)} = 1$.

Proof: Let us assume, for contradiction, that there exist optimal codeword lengths such that $\sum_{i=1}^k 2^{-\ell(\hat{x}_i)} < 1$. Then, $\theta = 0$ due to (5.11). From (5.9), we have

$$\hat{p}_i \ell(\hat{x}_i) + 2\hat{p}_i \mathbb{E}[L] - \lambda p_i = 0, \quad \forall i. \tag{5.16}$$

By summing over all i, we obtain $\mathbb{E}[L] = \frac{\lambda}{3}$. Then, we solve $\ell(\hat{x}_i) = \frac{\lambda}{3}$ for all i, which means $p(\lambda) = -\frac{\lambda^2}{9}$. By equating $p(\lambda)$ to zero, we obtain the optimal solution as $\lambda = 0$, which implies $\ell(\hat{x}_i) = 0$ for all i, which clearly cannot be a solution. Thus, we reach a contradiction, and the age-optimal codeword lengths must satisfy

$\sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} = 1$. ■

Due to Lemma 5.1, we have $\sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} = 1$ and $\theta \geq 0$. By summing (5.9) over all i, we obtain $\mathbb{E}[L]$ as

$$\mathbb{E}[L] = \frac{\lambda + \theta \log 2}{3}. \tag{5.17}$$

We rewrite (5.9), which is,

$$-\ell(\hat{x}_i) + \frac{\theta \log 2}{\hat{p}_i} 2^{-\ell(\hat{x}_i)} = 2\mathbb{E}[L] - \lambda \tag{5.18}$$

slightly differently as

$$\frac{\theta (\log 2)^2}{\hat{p}_i} 2^{-\ell(\hat{x}_i)} e^{\frac{\theta (\log 2)^2}{\hat{p}_i} 2^{-\ell(\hat{x}_i)}} = \frac{\theta (\log 2)^2}{\hat{p}_i} 2^{2\mathbb{E}[L]-\lambda}. \tag{5.19}$$

Note that (5.19) has the form of $xe^x = y$ for which the solution for x is equal to $x = W(y)$ if $y \geq 0$, where $W(\cdot)$ is the principle branch of the Lambert W function [150]. Using this, we find the unique solution for $\ell(\hat{x}_i)$ as

$$\ell(\hat{x}_i) = \left(\frac{\lambda - 2\theta \log 2}{3} \right) + \frac{1}{\log 2} W \left(\frac{\theta (\log 2)^2}{\hat{p}_i} 2^{\frac{-\lambda + 2\theta \log 2}{3}} \right). \tag{5.20}$$

We note that $\ell(\hat{x}_i)$ in (5.20) has two unknowns in it, θ and λ. In order to find the optimal codeword lengths, we find the (λ, θ) pair that satisfies $p(\lambda) = 0$ and $\sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} = 1$. Starting from an initial (λ, θ) pair, if $p(\lambda) > 0$ (or $p(\lambda) < 0$), we increase (or respectively decrease) λ in the next iteration as $p(\lambda)$ decreases in

λ. Next, we update θ by using (5.17). We repeat these steps until $p(\lambda) = 0$ and $\sum_{i=1}^{k} 2^{-\ell(\hat{x}_i)} = 1$.

5.3.2 Age-Optimal PMF for Given Codeword Lengths

In this section, we find the age-optimal pmf for given codeword lengths. For this case, we solve $\hat{p}_i$ as

$$\hat{p}_i = 2^{\frac{-\frac{1}{2}\ell(\hat{x}_i)^2 - 2\mathbb{E}[L]\ell(\hat{x}_i) + \lambda\ell(\hat{x}_i) - \sigma}{\gamma} - \frac{1}{\log 2}}. \tag{5.21}$$

In order to find the pmf for the partial updates, we solve (5.21) for a (γ, σ) pair that satisfies the entropy constraint $H(\hat{X}) = \beta$ and $\sum_{i=1}^{k} \hat{p}_i = 1$. Starting from an initial γ, if $H(\hat{X}) > \beta$ (or $H(\hat{X}) < \beta$), we increase (or respectively decrease) γ in the next iteration. Next, we update $\sigma = \frac{\gamma}{\log 2}(\log R - 1)$ where $R = \sum_{i=1}^{k} 2^{\frac{-\frac{1}{2}\ell(\hat{x}_i)^2 - 2\mathbb{E}[L]\ell(\hat{x}_i) + \lambda\ell(\hat{x}_i)}{\gamma}}$ to ensure $\sum_{i=1}^{k} \hat{p}_i = 1$. We repeat these steps until $H(\hat{X}) = \beta$ and $\sum_{i=1}^{k} \hat{p}_i = 1$.

5.3.3 The Overall Solution

Using an alternating minimization method [152–155], starting from an arbitrary pmf, we first find the age-optimal real-valued codeword lengths by following Section 5.3.1, and then update the pmf by following Section 5.3.2. We repeat this procedure until the first order optimality conditions in (5.9) and (5.10) are satisfied. Since the overall optimization problem in (5.7) is not convex, the solution obtained from this iterative alternating minimization algorithm may not be globally optimal.

Finally, recall that, in (5.7) and in Section 5.3.2, we solve for the pmf in an unconstrained manner. However, this k-point pmf must be such that it can be obtained from the original given n-point pmf by combining realizations. To solve the problem in (5.5) optimally, we need to search over all possible partitions of the original updates to generate the pmf for the partial updates, which is NP-hard. Instead, in order to solve this problem practically, especially for large n, we apply the proposed alternating minimization technique, which finds a proper pmf and the age-optimal codeword lengths. We then combine updates greedily to find a partition of the original updates that yields a pmf that approximates the solution obtained by the alternating minimization technique.

5.4 Numerical Results

We use Zipf(s, n) as the pmf for the original updates,

$$P_X(x_i) = \frac{i^{-s}}{\sum_{j=1}^{n} j^{-s}}, \quad i = 1, 2, \ldots, n. \tag{5.22}$$

For the first example, we use Zipf$(0.5, 8)$. We vary the entropy constraint β and find the corresponding optimum age with real-valued codeword lengths by searching over all possible non-empty partitions of the updates for $k \in \{3, 4, 5, 6\}$. We see in Fig. 5.3 that increasing the entropy constraint usually increases the average age. This is similar to classical compression [146] where entropy limits the minimum achievable average codeword length, i.e., $H(\hat{X}) \leq \mathbb{E}[L]$. Further, we observe in Fig. 5.3 that decreasing k achieves a lower average age for a given entropy constraint β. For example, when the entropy constraint is $\beta = 1.52$, the optimal age is equal

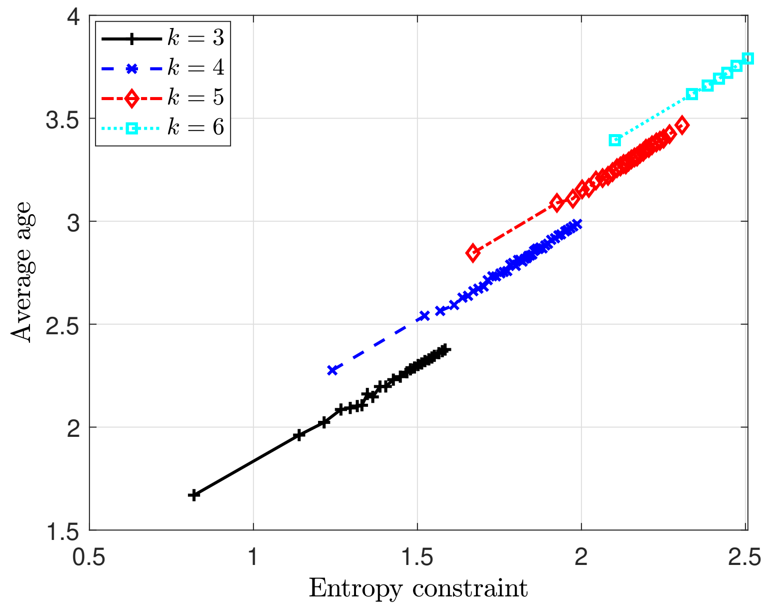
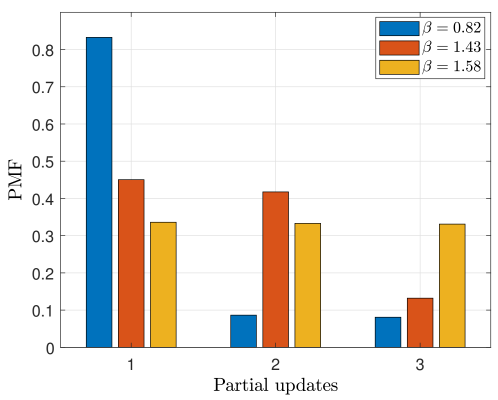


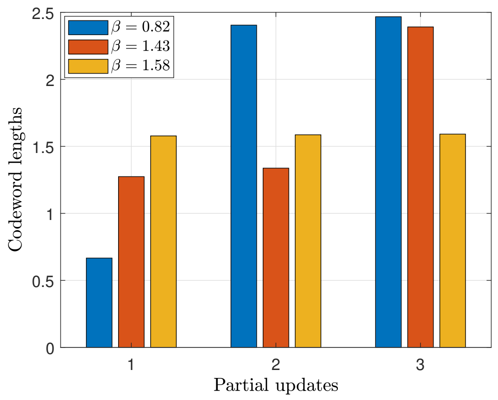
Figure 5.3: The optimum average age with real-valued codeword lengths when X is distributed with Zipf$(0.5, 8)$ for $k \in \{3, 4, 5, 6\}$.

to 2.54 for $k = 4$, whereas the optimal age is equal to 2.32 for $k = 3$.

For the second example, we again use Zipf$(0.5, 8)$ as the pmf for X. We find the age-optimal pmf in Fig. 5.4(a) and the corresponding age-optimal real-valued codeword lengths in Fig. 5.4(b) with respect to entropy constraint $\beta \in \{0.82, 1.43, 1.58\}$ when $k = 3$. We see that when the entropy constraint is relatively low, i.e., when $\beta = 0.82$, the age optimal partial updates are $\hat{\mathcal{X}} = \{\{x_1, x_2, \cdots, x_6\}, x_7, x_8\}$. In other words, the most probable six updates are mapped to one partial update, and the remaining two least probable updates are mapped to the other two partial updates. Since the most probable partial update has the smallest codeword length and realizes most frequently, the system achieves the lowest age compared to other possible partitions. When the entropy constraint is relatively high, i.e., when $\beta = 1.43$ and $\beta = 1.58$, we see in Fig. 5.4(a) that the optimum partial updates are



(a)



(b)

Figure 5.4: We find (a) the age-optimal pmf and (b) the corresponding age-optimal real-valued codeword lengths for $k = 3$ with respect to the entropy constraints $\beta \in \{0.82, 1.43, 1.58\}$ for an X with Zipf$(0.5, 8)$ distribution.

$\hat{\mathcal{X}} = \{\{x_2, x_4, x_6, x_8\}, \{x_1, x_5, x_7\}, x_3\}$ and $\hat{\mathcal{X}} = \{\{x_2, x_6, x_8\}, \{x_2, x_3, x_7\}, \{x_1, x_5\}\}$, respectively. Since entropy is a concave function of the pmf, when the entropy constraint is large, the optimum pmf for the partial updates gets closer to a uniform distribution. Thus, age-optimal partition policy strikes a balance between making some partial updates more probable and maintaining the entropy constraint.

In the first two examples, we used exhaustive search over partitions to obtain pmfs $\hat{p}$, and then for each pmf, we found the age-optimal codeword lengths using Section 5.3.1. For the third example, we use the proposed alternating minimization algorithm to find the pmf and the corresponding age-optimal codeword lengths which satisfy the first order optimality conditions in (5.9) and (5.10) for $k = 10$. We use the same initial pmf $P_{\hat{X}}(\hat{x}_i) = \{0.42, 0.32, 0.13, 0.1, 0.02, 0.007, 0.002, 0.0006, 0.00035, 0.00005\}$ for different entropy constraints. When the entropy constraint is relatively low, i.e., when $\beta = 1.6$, we see in Fig. 5.5(a) that the average age initially reduces

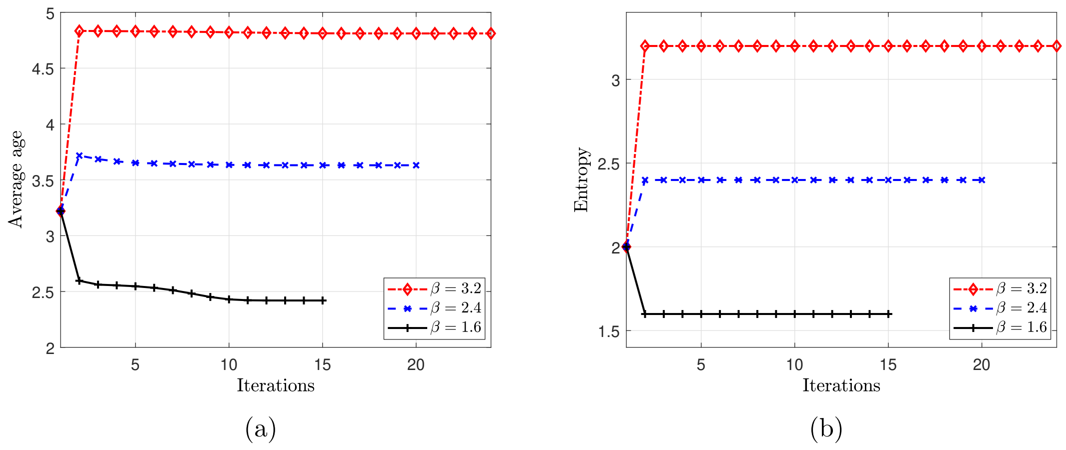


Figure 5.5: We use the proposed alternating minimization method to find the age-optimal pmf and the corresponding age-optimal real-valued codeword lengths for $k = 10$ with respect to the entropy constraints $\beta \in \{1.6, 2.4, 3.2\}$ starting from the same arbitrary pmf which has initial entropy close to 2. We show (a) the age evolution, and (b) entropy evolution, versus iteration index.

faster as the entropy reduces to the desired level 1.6. Then, the average age decreases over the remaining iterations. We find the desired pmf as $P_{\hat{X}}(\hat{x}_i) = \{0.3329,\ 0.3329,\ 0.3327,\ 0.0015,\ 0,\ \cdots,\ 0\}$ and its corresponding age-optimal lengths $\ell(\hat{x}_i) = \{1.59,\ 1.59,\ 1.59,\ 7.55,\ \infty,\ \cdots,\ \infty\}$. We note that even though $k = 10$, we see that the pmf has only five partial updates with positive probabilities. This is similar to the result in Fig. 5.3, i.e., decreasing k achieves lower average age with the same β. Further, the entropy in Fig. 5.5(b) remains the same after iteration two, as the algorithm always forces entropy condition to be satisfied with equality. When $\beta = 2.4$ and $\beta = 3.2$, the age increases initially as the entropy increases to the desired level. After the desired entropy is achieved, the average age decreases in the remaining steps. We find $P_{\hat{X}}(\hat{x}_i) = \{0.197,\ 0.197,\ 0.197,\ 0.197,\ 0.197,\ 0.015,\ 0,\ \cdots,\ 0\}$ and its corresponding age-optimal lengths $\ell(\hat{x}_i) = \{2.36,\ 2.36,\ 2.36,\ 2.36,\ 2.36,\ 5.23,$

$\infty, \cdots, \infty\}$ for $\beta = 2.4$. Further, for $\beta = 3.2$ we find $P_{\hat{X}}(\hat{x}_i) = \{0.1107, 0.1107, 0.1107, 0.1107, 0.1106, 0.1106, 0.1105, 0.1104, 0.1101, 0.005\}$ and its corresponding age-optimal lengths $\ell(\hat{x}_i) = \{3.18, 3.18, 3.18, 3.18, 3.181, 3.181, 3.182, 3.184, 3.188, 6.857\}$.

5.5 Conclusion

In this chapter, we studied the problem of generating partial updates, and finding their optimal real-valued codeword lengths to minimize the age of information at the receiver while maintaining a desired level of mutual information between the original and partial updates. As the original problem is NP hard, we relaxed this problem and proposed an alternating minimization based iterative algorithm that generates a pmf for the partial updates, and the corresponding age-optimal real-valued codeword length for each update. In numerical results, we observed the trade-off between obtaining more timely updates versus getting more informative updates at the receiver.

CHAPTER 6

Freshness in Cache Updating Systems

6.1 Introduction

In this chapter, we consider a cache updating system that consists of a source, cache(s) and user(s). We start with the simplest system model with a source, a single cache and a single end-user in Section 6.3 (and shown in Fig. 6.2); generalize it to the case where there are multiple caches in between the source and the user in Section 6.4 (and shown in Fig. 6.4); and further extend it to the case where there are multiple end-users in Section 6.7 (and shown in Fig. 6.6). The models we study are abstractions of a real-life setting shown in Fig. 6.1. Specifically, the two-hop serial cache system considered in Section 6.3 is an abstraction for the communication system between the cloud, macro base station and user A in Fig. 6.1; the multi-hop serial cache system considered in Section 6.4 is an abstraction for the communication system between the cloud, macro base station, small-cell base station and user B in Fig. 6.1 (for a three-hop system); and the multi-access caching system considered in Section 6.7 is an abstraction for the communication system between the cloud, macro base station, small-cell base station and users C and D in Fig. 6.1. Example

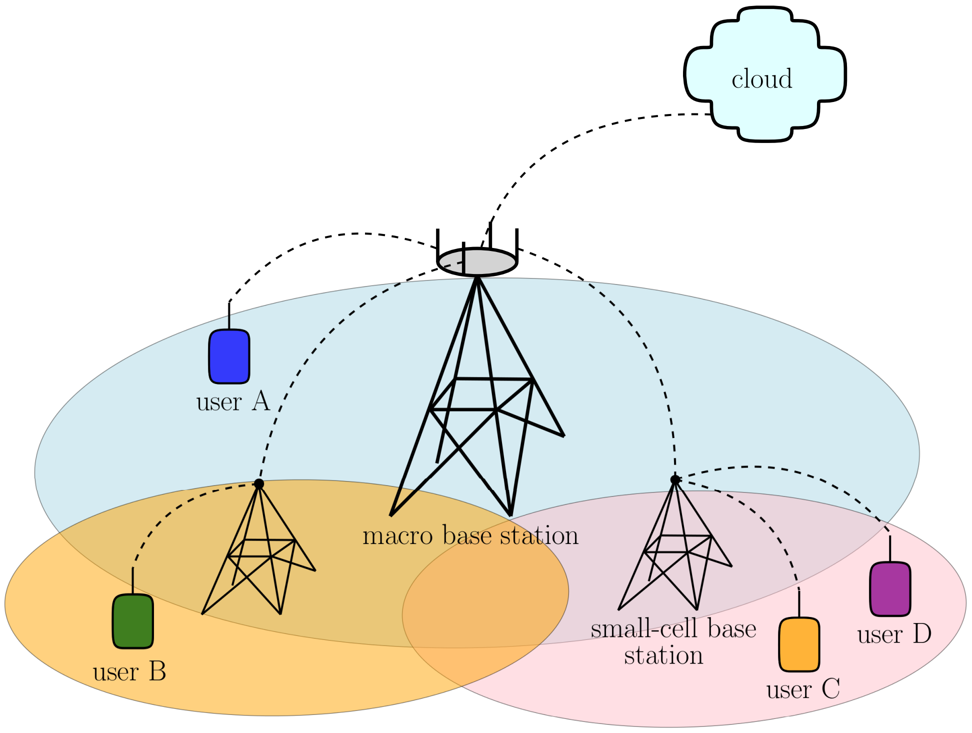


Figure 6.1: A cache updating system consisting of a cloud (the source), a macro base station (the first cache), a small-cell base station (the second cache), and users. The files at the source are updated with known rates. The first cache always obtains fresh files from the source. However, depending on the file status at the first cache, the second cache may not be able to obtain a fresh file all the time; the same is true for the users as well. We consider end-to-end freshness at the users.

deployment scenarios for hierarchical caching systems connected wirelessly can be found in 5G-enabled vehicular networks where self-sustaining wirelessly connected caching stations are placed to enhance vehicular network capacity [156].

In all these system models, the source keeps the freshest version of all the files which are updated with known rates λ_i. The cache downloads the files from the source and stores the latest downloaded versions of these files. When the cache downloads a file from the source, the file at the cache becomes fresh. After that, either the user gets the fresh file from the cache or the file at the cache becomes outdated due to a file update at the source. Thus, depending on the file status at the cache, the user may get a fresh or an outdated file. For all these system models, we derive analytical expressions for the information freshness at the end-users, and determine the updating frequencies for the intermediate caches and the end-users for maximum freshness.

References that are most closely related to our work are [28,93]. Reference [28] studies the problem of finding optimal crawl rates to keep the information in a search engine fresh while maintaining the constraints on crawl rates imposed by the websites and also the total crawl rate constraint of the search engine. Even though the freshness metric used in [28] is similar to ours, the problem settings are different where we develop a general freshness expression for a multi-hop multi-user caching system, which differentiates our overall work from [28]. Reference [93] considers a similar model to ours where a resource constrained remote server wants to keep the items at a local cache as fresh as possible. Reference [93] shows that the update rates of the files should be chosen proportional to the square roots of their popularity indices. Different from [93] where the freshness of the local cache is considered, we consider the freshness at the user which is connected to the source via a single cache or multiple caches. Thus, our system model can be thought of as an extended version of the one-hop model in [93]. However, our freshness metric is different than the traditional age metric used in [93], and hence, the overall work in this chapter is distinct compared to [93].

In this chapter, we first consider a system where there is a source, a cache and a user (Fig. 6.2). We find an analytical expression for the average freshness of the files at the user. We then generalize our result to find the average freshness for

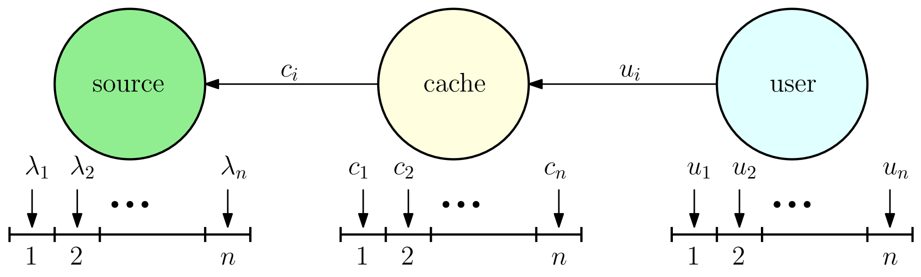


Figure 6.2: A cache updating system which consists of a source, a cache and a user. The ith file at the source is updated with rate λ_i, the cache requests updates for the ith file from the source with rate c_i, and the user requests updates for the ith file from the cache with rate u_i.

the end-user when multiple caches are placed in between the source and the user (Fig. 6.4). We impose total update rate constraints for the caches and also for the user due to limited nature of resources. Our aim is to find the update rates for the cache(s) and also for the user such that the total freshness of the files at the user is maximized. We find that the average freshness of the user is a concave function of the update rates of the caches and of the user individually, but not jointly. We provide an alternating maximization based solution where the update rates of the user (resp. of the cache) are optimized for a given set of update rates of the cache (resp. of the user). We observe that for a given set of parameters, such as update rates of the user, the optimal rate allocation policy for the other set of parameters, such as update rates at the caches, is a *threshold policy*, where the files that are updated frequently at the source may not be updated by the corresponding entity. Finally, we consider a system where multiple users are connected to a single cache (Fig. 6.6) and find update rates for the cache and for the users to maximize the total freshness over all users.

6.2 System Model, Freshness Function and Problem Formulation

We consider a cache updating system where there is an information source, a cache and a user as shown in Fig. 6.2. The information source keeps the freshest version of n files where the ith file is updated with exponential inter-arrival times with rate λ_i. The file updates at the source are independent of each other. A cache which is capable of storing the latest downloaded versions of all files gets the fresh files from the source. The channel between the source and the cache is assumed to be perfect and the transmission times are negligible, which is possible if the distance between the source and the cache and/or the file sizes are relatively small, as in [37,38,40–48]. Thus, if the cache requests an update for the ith file, it receives the file from the source right away. The inter-update request times of the cache for the ith file are exponential with rate c_i. The cache is subject to a total update rate constraint as in [93] as it is resource-constrained, i.e., $\sum_{i=1}^{n} c_i \leq C$.[1] The user requests the latest versions of the files stored in the cache. The inter-update request times of the user for the ith file are exponential with rate u_i. The channel between the user and the cache is also assumed to be perfect and the transmission times are negligible. Similarly, there is a total update rate constraint for the user, i.e., $\sum_{i=1}^{n} u_i \leq U$.

We note that each file at the source is always *fresh*. However, when a file is updated at the source, the stored versions of the same file at the cache and at the user become *outdated*. When the cache gets an update for an outdated file, the

[1] We note that in practical systems, the cache's frequent update requests from the source can cause network congestion problems. Even though we impose the total update rate constraint C due to the cache's limited resources, we can also imagine it to be imposed by the source in order to avoid network congestion problems.

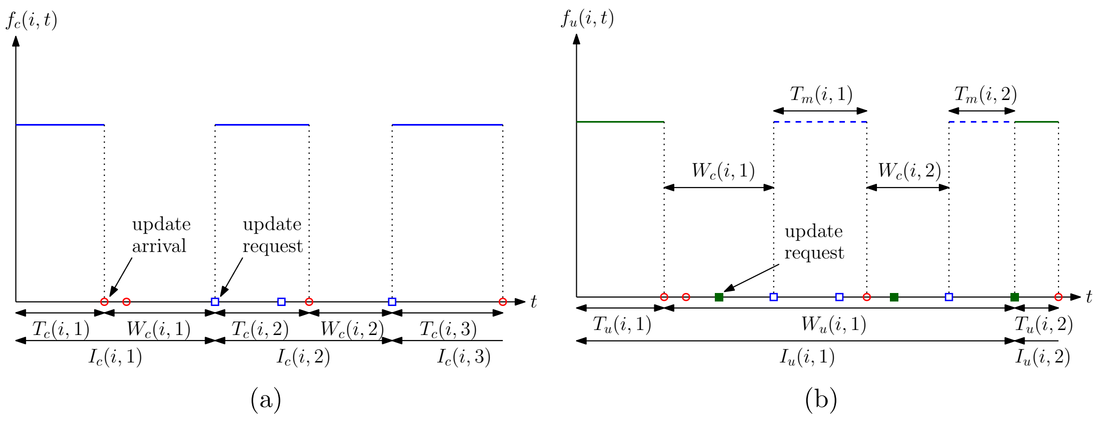


Figure 6.3: Sample evolution of freshness of the ith file (a) at the cache and (b) at the user. Red circles represent the update arrivals at the source, blue squares represent the update requests from the cache, and green filled squares represent the update requests from the user.

updated file in the cache becomes *fresh* again until the next update arrival at the source. When the user requests an update for an outdated file, it might still receive an outdated version if the file at the cache is not fresh. We note that since the cache and the user are unaware of the file updates at the source, they do not know whether they have the freshest versions of the files or not. Thus, they may still request an update even though they have the freshest version of a file.

In order to keep track of the *freshness*, we define the freshness function of the ith file at the cache $f_c(i,t)$ shown in Fig. 6.3(a) as,

$$f_c(i,t) = \begin{cases} 1, & \text{if the } i\text{th file is fresh at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

i.e., the instantaneous freshness function is a binary function taking values of fresh, “1”, or not fresh, “0”, at any time t. The binary freshness function [1] is different than the well-known age of information metric [2]. For example, if the information at the source does not change frequently, then the receiver may still have the freshest version of the information even though it has not received an update from the source for a while. In this case, the traditional age of information metric at the receiver would be high indicating that the information at the receiver is stale. Since the traditional age of information metric does not take into account the information change rate at the source, it may not be appropriate to measure information freshness in such systems. As the binary freshness metric compares the information at the receiver with the information stored at the source, it can measure information freshness at the receiver better than the traditional age of information in such settings.

We denote file updates which replace an outdated file with the freshest version of the file as successful updates. We define the time interval between the jth and the $(j+1)$th successful updates for the ith file at the cache as the jth update cycle and denote it by $I_c(i,j)$. We denote the time duration when the ith file at the cache is fresh during the jth update cycle as $T_c(i,j)$. Then, we define the long term average freshness of the ith file at the cache as

$$F_c(i) = \lim_{T\to\infty} \frac{1}{T} \int_0^T f_c(i,t)dt. \tag{6.2}$$

Let N denote the number of update cycles in the time duration T. Given that the system is ergodic, $\lim\limits_{T\to\infty} \frac{N}{T}$ exists and is finite. Then, similar to [1,2,10], $F_c(i)$ is equal to

$$F_c(i) = \lim_{T\to\infty} \frac{N}{T}\left(\frac{1}{N}\sum_{j=1}^{N} T_c(i,j)\right) = \frac{\mathbb{E}[T_c(i)]}{\mathbb{E}[I_c(i)]}. \tag{6.3}$$

Similarly, we define $f_u(i,t)$ as the freshness function of the ith file at the user shown in Fig. 6.3(b). Then, the long term average freshness of the ith file at the user is equal to

$$F_u(i) = \frac{\mathbb{E}[T_u(i)]}{\mathbb{E}[I_u(i)]}. \tag{6.4}$$

Finally, we define F_u as the total freshness over all files at the user as

$$F_u = \sum_{i=1}^{n} F_u(i). \tag{6.5}$$

Our aim is to find the optimal update rates for the cache, c_i, and for the user, u_i, for $i = 1, \dots, n$, such that the total average freshness of the user F_u is maximized while satisfying the constraints on the total update rate for the cache, $\sum_{i=1}^{n} c_i \leq C$, and for the user, $\sum_{i=1}^{n} u_i \leq U$. Thus, our optimization problem is,

$$\begin{aligned} \max_{\{c_i,u_i\}} \quad & F_u \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i \leq C \\ & \sum_{i=1}^{n} u_i \leq U \\ & c_i \geq 0, \quad u_i \geq 0, \quad i = 1, \dots, n. \end{aligned} \tag{6.6}$$

In the following section, we find analytical expressions for the long term average freshness of the ith file at the cache, $F_c(i)$, and at the user, $F_u(i)$, as a function of the update rate at the source λ_i, the update rate at the cache c_i, and the update rate at the user u_i. Once we find $F_u(i)$, this will determine the objective function of (6.6) via (6.5).

6.3 Average Freshness Analysis for a Single Cache

In this section, we consider the system model in Fig. 6.2, where there is a source, a single cache and a user. First, we find an analytical expression for the long term average freshness of the ith file at the cache, i.e., $F_c(i)$ in (6.3). We note that due to the memoryless property of the exponential distribution, $T_c(i,j)$ which is the time duration when the ith file at the cache is fresh during the jth update cycle is exponentially distributed with parameter λ_i. Since $T_c(i,j)$ are independent and identically distributed (i.i.d.) over j, we drop index j, and denote a typical $T_c(i,j)$ as $T_c(i)$. Thus, we have $\mathbb{E}[T_c(i)] = \frac{1}{\lambda_i}$. Let $W_c(i,j)$ be the total duration when the ith file at the cache is outdated during the jth update cycle, i.e., $W_c(i,j) = I_c(i,j) - T_c(i,j)$. Note that $W_c(i,j)$ is also equal to the time passed until the fresh version of the ith file is obtained from the source after the file is outdated at the cache. We denote typical random variables for $W_c(i,j)$ and $I_c(i,j)$ by $W_c(i)$ and $I_c(i)$, respectively. As the update request times for the cache are exponentially distributed with rate c_i, we have $\mathbb{E}[W_c(i)] = \frac{1}{c_i}$. Thus, we find

$$\mathbb{E}[I_c(i)] = \mathbb{E}[T_c(i)] + \mathbb{E}[W_c(i)] = \frac{1}{\lambda_i} + \frac{1}{c_i}. \tag{6.7}$$

By using (6.3), we find $F_c(i)$ as

$$F_c(i) = \frac{c_i}{c_i + \lambda_i}. \tag{6.8}$$

We note that the freshness of the ith file at the cache $F_c(i)$ in (6.8) is an increasing function of the cache update rate c_i, but a decreasing function of the source update rate λ_i.[2]

Next, we find an analytical expression for the average freshness of the ith file at the user $F_u(i)$. Similar to $\mathbb{E}[T_c(i)]$, we have $\mathbb{E}[T_u(i)] = \frac{1}{\lambda_i}$ due to the memoryless property of the exponential distribution, i.e., after the user gets the fresh file, the remaining time for the next file update at the source is still exponentially distributed with rate λ_i. Similarly, we denote the time duration when the ith file at the user is outdated during the jth update cycle as $W_u(i,j)$ which is equal to $W_u(i,j) = I_u(i,j) - T_u(i,j)$. In order for the user to get fresh updates from the cache, the cache needs to get the fresh update from the source which takes $W_c(i,j)$ time as

[2]We note that [1] investigates several updating policies, including fixed-order updating, random-order updating, and purely random updating. The freshness metric with fixed-order updating is $F_{\text{fixed-order}} = \frac{c_i}{\lambda_i}\left(1 - e^{-\frac{\lambda_i}{c_i}}\right)$, with random-order updating is $F_{\text{random-order}} = \frac{c_i}{\lambda_i}\left(1 - \frac{c_i^2}{\lambda_i^2}\left(1 - e^{-\frac{\lambda_i}{c_i}}\right)^2\right)$, and with purely random updating is $F_{\text{purely-random}} = \frac{c_i}{c_i+\lambda_i}$, as given in (6.8). All these three functions are monotonically decreasing in $\frac{\lambda_i}{c_i}$, and have similar forms when plotted. We adopt purely random updating in this chapter due to its simplicity, and amenability to yield closed form expressions when optimized.

discussed earlier. After the file at the cache becomes fresh, either the user gets the fresh update from the cache or the file at the source is updated, and thus the file at the cache becomes outdated again. We denote the earliest time that one of these two cases happens as $T_m(i)$, i.e., $T_m(i) = \min\{T_c(i), \bar{W}_u(i)\}$ where $\bar{W}_u(i)$ is the time for the user to obtain a new update from the cache which is exponentially distributed with rate u_i. Thus, $T_m(i)$ is also exponentially distributed with rate $u_i + \lambda_i$. We note that $\mathbb{P}[T_m(i) = \bar{W}_u(i)] = \frac{u_i}{u_i+\lambda_i}$ and $\mathbb{P}[T_m(i) = T_c(i)] = \frac{\lambda_i}{u_i+\lambda_i}$.

Note that if the user gets the fresh update before the file at the cache becomes outdated which happens with probability $\mathbb{P}[T_m(i) = \bar{W}_u(i)]$, an update cycle of the ith file at the user is completed and thus, $f_u(i,t)$ is equal to 1 again. However, if the file at the source is updated before the user gets the fresh update from the cache, then this process repeats itself, i.e., the cache initially needs to receive the fresh update which takes another $W_c(i,j)$ time and so on, until the user receives the fresh update from the cache. Thus, we write $W_u(i,j)$ as

$$W_u(i,j) = \sum_{k=1}^{K} W_c(i,k) + T_m(i,k), \tag{6.9}$$

where K is a geometric random variable with rate $\frac{u_i}{u_i+\lambda_i}$. Due to [147, Prob. 9.4.1], $\sum_{k=1}^{K} W_c(i,k)$ and $\sum_{k=1}^{K} T_m(i,k)$ are exponentially distributed with rates $\frac{u_i c_i}{u_i+\lambda_i}$ and u_i, respectively. We use $W_u(i)$ and $I_u(i)$ to denote the typical random variables for $W_u(i,j)$ and $I_u(i,j)$, respectively. Thus, we have $\mathbb{E}[W_u(i)] = \frac{u_i+\lambda_i}{u_i c_i} + \frac{1}{u_i}$. Since

$\mathbb{E}[I_u(i)] = \mathbb{E}[T_u(i)] + \mathbb{E}[W_u(i)]$, we get

$$\mathbb{E}[I_u(i)] = \frac{1}{\lambda_i} + \frac{1}{u_i} + \frac{u_i + \lambda_i}{u_i c_i}. \tag{6.10}$$

Finally, we find $F_u(i)$ as

$$F_u(i) = \frac{\mathbb{E}[T_u(i)]}{\mathbb{E}[I_u(i)]} = \frac{u_i}{u_i + \lambda_i} \frac{c_i}{c_i + \lambda_i}. \tag{6.11}$$

We note that the freshness of the ith file at the user $F_u(i)$ in (6.11) depends not only on the update rate of the user u_i and file update rate at the source λ_i but also the update rate of the cache c_i as the user obtains the fresh update from the cache.[3] [4] We note that $F_u(i)$ is an increasing function of u_i and c_i, but a decreasing function of λ_i. We observe that $F_u(i)$ is an individually concave function of u_i and c_i but not jointly concave in u_i and c_i, as u_i and c_i terms appear as a multiplication in (6.11). If the user was directly connected to the source, its freshness would be equal to $\frac{u_i}{u_i+\lambda_i}$, i.e., the first term in (6.11). However, as the user is connected to the source

[3]In this chapter, we assume that the channel is perfect and the transmission times are negligible. We note that $F_u(i)$ in (6.11) can be extended to a case where the transmissions are still instantaneous but are not error free. Let p_i and q_i be the probabilities of successfully transmitting the ith file for the cache and for the user, respectively. By using [68, Prob. 9.4.1], one can show that the successful inter-update request times for the cache and for the user are exponentially distributed with rates $p_i c_i$ and $q_i u_i$, respectively. Then, the freshness of the ith file at the user becomes $F_u(i) = \frac{q_i u_i}{q_i u_i + \lambda_i} \frac{p_i c_i}{p_i c_i + \lambda_i}$. We observe that the freshness of the ith file at the user increase with p_i and q_i as expected.

[4]Even though we neglect the transmission times, as an extension to our work, in [99], we consider a cache updating system where the user is able to obtain uncached files directly from the source. However, the channel between the user and the source is imperfect, and thus there is a transmission time which is exponentially distributed with rate s_i. In [99], if the ith file is not cached and the cache is able to send all the file update requests from the user to the source, then the freshness of the ith file is equal to $F_u(i) = \frac{u_i}{u_i+\lambda_i+\frac{u_i\lambda_i}{s_i}}$. Thus, we observe that $F_u(i)$ decreases with the transmission delays as the additional term $\frac{u_i\lambda_i}{s_i}$ in the denominator decreases the freshness.

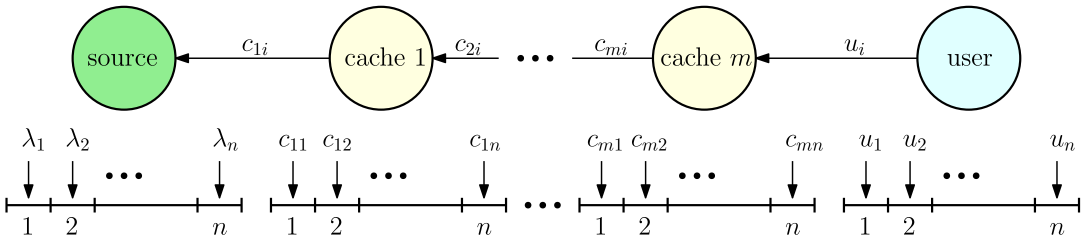


Figure 6.4: Generalized system model where there are m serially connected caches in between the source and the user.

via the cache, the freshness experienced by the user is equal to the multiplication of the freshness of the cache and the freshness of the user if the user was directly connected to the source. Note that, since $\frac{c_i}{c_i+\lambda_i} < 1$, the freshness of the user when connected to the source via a cache is smaller than the freshness it would achieve if it was directly connected to the source.

In the following section, we find the average freshness of the caches and of the user for the general case when there are m caches connected serially in between the source and the user.

6.4 Average Freshness Analysis for M Caches

In this section, we consider a system where there are m caches placed in between the source and the user, as shown in Fig. 6.4. We denote the rth cache's update rate for the ith file as c_{ri}. We define $I_c(r,i,j)$ as the jth update cycle for the ith file at cache r for $r = 1,\dots,m$. Similarly, we define $T_c(r,i,j)$ (and $W_c(r,i,j)$) as the time duration when the ith file at cache r is fresh (and outdated) during the jth update cycle, i.e., we have $I_c(r,i,j) = T_c(r,i,j) + W_c(r,i,j)$.

Next, we find an analytical expression for the average freshness of the ith file at the rth cache $F_c(r,i)$ and at the user $F_u(i)$. In order to obtain a fresh file at cache r, the file at cache $r-1$ needs to be fresh for $r>1$. Similar to the derivation of $F_u(i)$ in (6.11), after cache $r-1$ obtains the fresh file, either cache r gets the fresh file from cache $r-1$ or the file at the source is updated and the file in all the caches becomes outdated. Thus, we write $W_c(r,i,j)$ as

$$W_c(r,i,j) = \sum_{\ell=1}^{K_r} W_c(r-1,i,\ell) + T_m(r,i,\ell), \tag{6.12}$$

where K_r is a geometric random variable with rate $\frac{c_{ri}}{c_{ri}+\lambda_i}$ and $T_m(r,i) = \min\{T_c(r,i), \bar{W}_c(r,i)\}$ where $T_c(r,i)$ and $\bar{W}_c(r,i)$ are exponentially distributed with rates λ_i and c_{ri}, respectively. We note that $T_m(r,i)$ is also exponentially distributed with rate $c_{ri}+\lambda_i$. Then, given that $K_r = k$, we write $\mathbb{E}[W_c(r,i)|K_r=k]$ as

$$\mathbb{E}[W_c(r,i)|K_r=k] = k\left(\mathbb{E}[W_c(r-1,i)] + \frac{1}{c_{ri}+\lambda_i}\right). \tag{6.13}$$

Then, we find $\mathbb{E}[W_c(r,i)] = \mathbb{E}\left[\mathbb{E}\left[W_c(r,i)|K_r\right]\right]$ as

$$\mathbb{E}[W_c(r,i)] = \frac{c_{ri}+\lambda_i}{c_{ri}}\mathbb{E}[W_c(r-1,i)] + \frac{1}{c_{ri}}, \tag{6.14}$$

which is equal to $\mathbb{E}[W_c(1,i)] = \frac{1}{c_{1i}}$ if $r=1$, and to

$$\mathbb{E}[W_c(r,i)] = \frac{1}{c_{ri}} + \sum_{\ell=1}^{r-1} \frac{1}{c_{\ell i}} \prod_{p=\ell+1}^{r} \frac{c_{pi}+\lambda_i}{c_{pi}}, \quad r = 2,\dots,m. \tag{6.15}$$

Then, by using $\mathbb{E}[I_c(r,i)] = \mathbb{E}[T_c(r,i)] + \mathbb{E}[W_c(r,i)]$, we have

$$\mathbb{E}[I_c(r,i)] = \begin{cases} \frac{1}{\lambda_i} + \frac{1}{c_{1i}}, & r = 1, \\ \left(\frac{1}{\lambda_i} + \frac{1}{c_{1i}}\right) \prod_{\ell=2}^{r} \frac{c_{\ell i}+\lambda_i}{c_{\ell i}}, & r = 2, \dots, m. \end{cases} \tag{6.16}$$

Finally, we find the average freshness for the ith file at cache r as

$$F_c(r,i) = \frac{\mathbb{E}[T_c(r,i)]}{\mathbb{E}[I_c(r,i)]} = \prod_{\ell=1}^{r} \frac{c_{\ell i}}{c_{\ell i}+\lambda_i}, \quad r = 1, \dots, m. \tag{6.17}$$

Similarly, we find $\mathbb{E}[I_u(i)]$ as

$$\mathbb{E}[I_u(i)] = \left(\frac{1}{\lambda_i} + \frac{1}{c_{1i}}\right) \frac{u_i+\lambda_i}{u_i} \prod_{r=2}^{m} \frac{c_{ri}+\lambda_i}{c_{ri}}. \tag{6.18}$$

Then, the average freshness of the ith file at the user is

$$F_u(i) = \frac{\mathbb{E}[T_u(i)]}{\mathbb{E}[I_u(i)]} = \frac{u_i}{u_i+\lambda_i} \prod_{r=1}^{m} \frac{c_{ri}}{c_{ri}+\lambda_i}. \tag{6.19}$$

Thus, we observe from (6.17) that, for the general system, the average freshness experienced by cache r for $r > 1$ is equal to the multiplication of the freshness of cache $r-1$ with the freshness of cache r when cache r is directly connected to the source. We observe from (6.19) that the same structure is valid for the freshness of the user as well. We also note that the average freshness expression in (6.19) reduces to the expression in (6.11) found in Section 6.3, when $m = 1$. Finally, as an explicit example of the expression in (6.19), if we have $m = 2$ caches between the source and the user, the freshness at the user is

$$F_u(i) = \frac{u_i}{u_i+\lambda_i}\frac{c_{1i}}{c_{1i}+\lambda_i}\frac{c_{2i}}{c_{2i}+\lambda_i}. \tag{6.20}$$

In the following section, we solve the optimization problem in (6.6) for the system with a single cache by using the freshness expression $F_u(i)$ found in (6.11) in Section 6.3.

6.5 Freshness Maximization for a System with a Single Cache

In this section, we consider the optimization problem in (6.6) for a system with a single cache. Using $F_u(i)$ in (6.11) and F_u in (6.5), we rewrite the freshness maximization problem as

$$\begin{aligned} \max_{\{c_i,u_i\}} \quad & \sum_{i=1}^{n} \frac{u_i}{u_i+\lambda_i}\frac{c_i}{c_i+\lambda_i} \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i \leq C \\ & \sum_{i=1}^{n} u_i \leq U \\ & c_i \geq 0, \quad u_i \geq 0, \quad i=1,\dots,n. \end{aligned} \tag{6.21}$$

We introduce the Lagrangian function [149] for (6.21) as

$$\mathcal{L} = -\sum_{i=1}^{n} \frac{u_i}{u_i+\lambda_i}\frac{c_i}{c_i+\lambda_i} + \beta\left(\sum_{i=1}^{n} c_i - C\right) + \theta\left(\sum_{i=1}^{n} u_i - U\right) - \sum_{i=1}^{n}\nu_i c_i - \sum_{i=1}^{n}\eta_i u_i, \tag{6.22}$$

where $\beta \geq 0$, $\theta \geq 0$, $\nu_i \geq 0$ and $\eta_i \geq 0$. Then, we write the KKT conditions as

$$\frac{\partial \mathcal{L}}{\partial c_i} = -\frac{u_i}{u_i + \lambda_i} \frac{\lambda_i}{(c_i + \lambda_i)^2} + \beta - \nu_i = 0, \tag{6.23}$$

$$\frac{\partial \mathcal{L}}{\partial u_i} = -\frac{c_i}{c_i + \lambda_i} \frac{\lambda_i}{(u_i + \lambda_i)^2} + \theta - \eta_i = 0, \tag{6.24}$$

for all i. The complementary slackness conditions are

$$\beta \left(\sum_{i=1}^{n} c_i - C \right) = 0, \tag{6.25}$$

$$\theta \left(\sum_{i=1}^{n} u_i - U \right) = 0, \tag{6.26}$$

$$\nu_i c_i = 0, \tag{6.27}$$

$$\eta_i u_i = 0. \tag{6.28}$$

The objective function in (6.21) is not jointly concave in c_i and u_i since c_is and u_is appear as multiplicative terms in the objective function. However, for given c_is, the objective function in (6.21) is concave in u_i. Similarly, for given u_is, the objective function in (6.21) is concave in c_i. Thus, we apply an alternating maximization based method [87, 152–154] to find (c_i, u_i) pairs such that (6.23) and (6.24) are satisfied for all i.[5]

Starting with initial u_is, we find the optimum update rates for the cache, c_is, such that the total update rate constraint for the cache, i.e., $\sum_{i=1}^{n} c_i \leq C$, and the

[5]The proposed alternating maximization based method finds (c_i, u_i) pairs that satisfy the first order optimality conditions, i.e., the KKT conditions in (6.23)-(6.28). We note that as the optimization problem in (6.21) is not a convex optimization problem, the solutions obtained from the alternating maximization based method are not globally optimal. Similarly, solutions obtained for the optimization problems in (6.37) and (6.51) are locally optimal as well.

feasibility of the update rates, i.e., $c_i \geq 0$ for all i, are satisfied. Then, for given c_is, we find the optimum update rates for the user, u_is, such that the total update rate constraint for the user, i.e., $\sum_{i=1}^{n} u_i \leq U$, and the feasibility of the update rates, i.e., $u_i \geq 0$ for all i, are satisfied. We repeat these steps until the KKT conditions in (6.23) and (6.24) are satisfied.

For given u_is with $u_i > 0$, we rewrite (6.23) as

$$(c_i + \lambda_i)^2 = \frac{1}{\beta - \nu_i} \frac{u_i \lambda_i}{u_i + \lambda_i} \tag{6.29}$$

Then, we find c_i as

$$c_i = \frac{1}{\sqrt{\beta - \nu_i}} \sqrt{\frac{u_i \lambda_i}{u_i + \lambda_i}} - \lambda_i, \tag{6.30}$$

for all i with $u_i > 0$. If $c_i > 0$, we have $\nu_i = 0$ from (6.27). Thus, we have

$$c_i = \left(\frac{1}{\sqrt{\beta}} \sqrt{\frac{u_i \lambda_i}{u_i + \lambda_i}} - \lambda_i \right)^+, \tag{6.31}$$

for all i with $u_i > 0$, where $(x)^+ = \max(x, 0)$. Note that $c_i > 0$ requires $\frac{1}{\lambda_i} \frac{u_i}{u_i + \lambda_i} > \beta$ which also implies that if $\frac{1}{\lambda_i} \frac{u_i}{u_i + \lambda_i} \leq \beta$, then we must have $c_i = 0$. Thus, for given u_is, we see that the optimal rate allocation policy for the cache is a *threshold policy* in which the optimal update rates are equal to zero when the files are updated too frequently at the source, i.e., when the corresponding λ_is are too large.[6] [7]

Next, we solve for u_is for given c_is with $c_i > 0$. We rewrite (6.24) as

$$(u_i + \lambda_i)^2 = \frac{1}{\theta - \eta_i} \frac{c_i \lambda_i}{c_i + \lambda_i}. \tag{6.32}$$

Then, we find u_i as

$$u_i = \frac{1}{\sqrt{\theta - \eta_i}} \sqrt{\frac{c_i \lambda_i}{c_i + \lambda_i}} - \lambda_i, \tag{6.33}$$

for all i with $c_i > 0$. If $u_i > 0$, we have $\eta_i = 0$ from (6.28). Thus, we have

$$u_i = \left(\frac{1}{\sqrt{\theta}} \sqrt{\frac{c_i \lambda_i}{c_i + \lambda_i}} - \lambda_i \right)^+. \tag{6.34}$$

Similarly, $u_i > 0$ requires $\frac{1}{\lambda_i} \frac{c_i}{c_i + \lambda_i} > \theta$ which implies that if $\frac{1}{\lambda_i} \frac{c_i}{c_i + \lambda_i} \leq \theta$, then we must have $u_i = 0$. Thus, for given c_is, we see that the optimal rate allocation policy for the user is also a threshold policy in which the optimal update rates are equal to zero when the files are updated too frequently at the source, i.e., when the corresponding λ_is are too large.

In the following lemma, we show that if the update rate of the cache c_i (resp.

[6] As the stored versions of the files that change too frequently at the source become obsolete too quickly at the user, the freshness of these files at the user will be small. That is why, instead of updating these files, with the threshold policy, the files that change less frequently at the source are updated more which brings higher contribution to the overall freshness at the user.

[7] As a result of the threshold policy, the user may not receive the fresh versions of the files that change too frequently at the source which can be undesirable especially if obtaining the fresh versions of some files is more important than the others. In order to address this problem, we can introduce an importance factor for each file μ_i. Then, we can rewrite the overall freshness at the user F_u in (6.5) as $F_u = \sum_{i=1}^{n} \mu_i F_u(i)$. We note that by solving the optimization problem in (6.6) with a freshness expression with importance factors, the important files that change too frequently at the source may be updated by the cache and also by the user.

of the user u_i) is equal to zero for the ith file, then the update rate of the user u_i (resp. of the cache c_i) for the same file must also be equal to zero.

Lemma 6.1 *In the optimal policy, if $c_i = 0$, then we must have $u_i = 0$; and vice versa.*

Proof: Assume for contradiction that in the optimal policy, there exist update rates with $c_i = 0$ and $u_i > 0$. We note that average freshness of this file at the user is equal to zero, i.e., $F_u(i) = 0$, as $c_i = 0$. We can increase the total freshness of the user F_u by decreasing u_i to zero and increasing one of u_js with $c_j > 0$. Thus, we reach a contradiction and in the optimal policy, if $c_i = 0$, we must have $u_i = 0$. For the update rates with $c_i > 0$ and $u_i = 0$, one can similarly show that if $u_i = 0$, then we must have $c_i = 0$. ■

In the following lemma, we show that the total update rate constraints for the cache, i.e., $\sum_{i=1}^n c_i \leq C$, and for the user, i.e., $\sum_{i=1}^n u_i \leq U$, must be satisfied with equality.

Lemma 6.2 *In the optimal policy, we must have $\sum_{i=1}^n c_i = C$ and $\sum_{i=1}^n u_i = U$.*

Proof: Assume for contradiction that in the optimal policy, we have $\sum_{i=1}^n c_i < C$. As the objective function in (6.21) is an increasing function of c_i, we can increase the total freshness of the user F_u by increasing one of c_i with $u_i > 0$ until the total update rate constraint for the cache is satisfied with equality, i.e., $\sum_{i=1}^n c_i = C$. Thus, we reach a contradiction and in the optimal policy, we must have $\sum_{i=1}^n c_i = C$. By using a similar argument, we can also show that in the optimal policy, we must have

$\sum_{i=1}^{n} u_i = U$. ■

In the following lemma, we identify a property of the optimal cache update rates c_i for given user update rates u_i. To that end, for given u_is, let us define ϕ_is as

$$\phi_i = \frac{1}{\lambda_i}\frac{u_i}{u_i+\lambda_i}. \tag{6.35}$$

This lemma will be useful for solving for c_i given u_i using (6.31).

Lemma 6.3 *For given u_is, if $c_i > 0$ for some i, then we have $c_j > 0$ for all j with $\phi_j \geq \phi_i$.*

Proof: As we noted earlier, from (6.31), $c_i > 0$ implies $\phi_i > \beta$. Thus, if $\phi_j \geq \phi_i$, then we have $\phi_j > \beta$, which further implies $c_j > 0$. ■

Next, we describe the overall solution for the single cache setting. We start with a set of initial u_is. We obtain ϕ_i from u_i using (6.35). We will utilize Fig. 6.5 to describe the steps of the solution visually. We plot ϕ_i in Fig. 6.5. Note that if $u_i = 0$ then $\phi_i = 0$, and vice versa. First, we choose $c_i = 0$ for the files with $u_i = 0$ due to Lemma 6.1, i.e., in Fig. 6.5, we choose c_3 and c_6 as zero. Next, we find the remaining c_is with $u_i > 0$. For that, we rewrite (6.31) as

$$c_i = \frac{\lambda_i}{\sqrt{\beta}}\left(\sqrt{\phi_i}-\sqrt{\beta}\right)^{+}. \tag{6.36}$$

Due to Lemma 6.2, in the optimal policy, we must have $\sum_{i=1}^{n} c_i = C$. Assuming that $\phi_i \geq \beta$ for all i, i.e., by ignoring $(\cdot)^+$ in (6.31) and (6.36), we solve $\sum_{i=1}^{n} c_i = C$

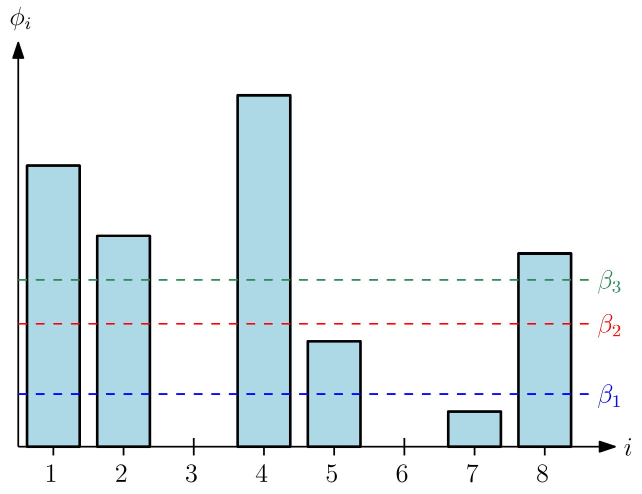


Figure 6.5: For given u_is, we show ϕ_is calculated in (6.35) for $n = 8$.

for β. Then, we compare the smallest ϕ_i with β. If the smallest ϕ_i is larger than or equal to β, it implies that $c_i > 0$ for all i due to Lemma 6.3, and we have obtained the optimal c_i values for given u_is. If the smallest ϕ_i is smaller than β, it implies that the corresponding c_i was negative and it must be chosen as zero. In this case, we choose $c_i = 0$ for the smallest ϕ_i. In the example in Fig. 6.5, if the β we found is β_1, then since $\phi_7 < \beta_1$ we choose c_7 as zero. Then, we repeat this process again until the smallest ϕ_i among the remaining c_is satisfies $\phi_i \geq \beta$. For example, in Fig. 6.5, the next β found by using only indices $1, 2, 4, 5, 8$ may be β_2. Since $\phi_5 < \beta_2$, we choose $c_5 = 0$. In the next iteration, the β found by using indices $1, 2, 4, 8$ may be β_3. Since $\phi_8 > \beta_3$, we stop the process and find c_i for $i = 1, 2, 4, 8$ from (6.31) or (6.36) by using β_3 in Fig. 6.5. This concludes finding c_is for given u_is. Next, for given c_is, we find u_is by following a similar procedure. We keep solving for c_is for given u_is, and u_is for given c_is, until (c_i, u_i) pairs converge.

In the following section, we provide a solution for the general system with multiple caches.

6.6 Freshness Maximization for the General System

In this section, we provide a solution for the general system shown in Fig. 6.4, where there are m caches in between the source and the user. We define C_r as the total update rate of cache r, i.e., $\sum_{i=1}^{n} c_{ri} \leq C_r$. Using $F_u(i)$ in (6.19), we rewrite the optimization problem in (6.6) as

$$\begin{aligned} \max_{\{c_{ri},u_i\}} \quad & \sum_{i=1}^{n} \frac{u_i}{u_i+\lambda_i} \prod_{r=1}^{m} \frac{c_{ri}}{c_{ri}+\lambda_i} \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_{ri} \leq C_r, \quad r=1,\dots,m \\ & \sum_{i=1}^{n} u_i \leq U \\ & c_{ri} \geq 0, \quad u_i \geq 0, \quad r=1,\dots,m, \quad i=1,\dots,n. \end{aligned} \tag{6.37}$$

We introduce the Lagrangian function for (6.37) as

$$\begin{aligned} \mathcal{L} = & -\sum_{i=1}^{n} \frac{u_i}{u_i+\lambda_i} \prod_{r=1}^{m} \frac{c_{ri}}{c_{ri}+\lambda_i} + \sum_{r=1}^{m} \beta_r \left(\sum_{i=1}^{n} c_{ri} - C_r \right) + \theta \left(\sum_{i=1}^{n} u_i - U \right) \\ & - \sum_{r=1}^{m} \sum_{i=1}^{n} \nu_{ri} c_{ri} - \sum_{i=1}^{n} \eta_i u_i, \end{aligned} \tag{6.38}$$

where $\beta_r \geq 0$, $\theta \geq 0$, $\nu_{ri} \geq 0$ and $\eta_i \geq 0$. Then, we write the KKT conditions as

$$\frac{\partial \mathcal{L}}{\partial c_{ri}} = -\frac{u_i}{u_i+\lambda_i} \prod_{\ell \neq r} \frac{c_{\ell i}}{c_{\ell i}+\lambda_i} \frac{\lambda_i}{(c_{ri}+\lambda_i)^2} + \beta_r - \nu_{ri} = 0, \tag{6.39}$$

$$\frac{\partial \mathcal{L}}{\partial u_i} = -\frac{\lambda_i}{(u_i+\lambda_i)^2} \prod_{r=1}^{m} \frac{c_{ri}}{c_{ri}+\lambda_i} + \theta - \eta_i = 0, \tag{6.40}$$

for all r and i. The complementary slackness conditions are

$$\beta_r \left(\sum_{i=1}^{n} c_{ri} - C_r \right) = 0, \tag{6.41}$$

$$\theta \left(\sum_{i=1}^{n} u_i - U \right) = 0, \tag{6.42}$$

$$\nu_{ri} c_{ri} = 0, \tag{6.43}$$

$$\eta_i u_i = 0. \tag{6.44}$$

The objective function in (6.37) is not jointly concave in c_{ri} and u_i. However, for given c_{ri}s, the objective function in (6.37) is concave in u_i, and for a given u_i and $c_{\ell i}$s for all $\ell \neq r$, the objective function in (6.37) is concave in c_{ri}. Thus, similar to the solution approach in Section 6.5, we apply an alternating maximization based method to find $(c_{1i}, \dots, c_{ri}, u_i)$ tuples such that (6.39) and (6.40) are satisfied for all r and i.

Starting with initial u_i and $c_{\ell i}$s for $\ell \neq r$, we find the optimum update rates for cache r, c_{ri}s, such that the total update rate constraint for the cache, i.e., $\sum_{i=1}^{n} c_{ri} \leq C_r$, and the feasibility of the update rates, i.e., $c_{ri} \geq 0$ for all i, are satisfied. We repeat this step for all r. Then, for given c_{ri}s, we find the optimum update rates for the user, u_is, such that the total update rate constraint for the user, i.e., $\sum_{i=1}^{n} u_i \leq U$, and the feasibility of the update rates, i.e., $u_i \geq 0$ for all i, are satisfied. We repeat these steps until the KKT conditions in (6.39) and (6.40) are satisfied.

For given u_is with $u_i > 0$, and $c_{\ell i}$ with $c_{\ell i} > 0$, for $\ell \neq r$, we rewrite (6.39) as

$$(c_{ri} + \lambda_i)^2 = \frac{\sigma_i \lambda_i}{\beta_r - \nu_{ri}}, \tag{6.45}$$

where $\sigma_i = \frac{u_i}{u_i+\lambda_i} \prod_{\ell \neq r} \frac{c_{\ell i}}{c_{\ell i}+\lambda_i}$. Then, we find c_{ri} as

$$c_{ri} = \sqrt{\frac{\sigma_i \lambda_i}{\beta_r - \nu_{ri}}} - \lambda_i, \tag{6.46}$$

for all i with $u_i > 0$ and $c_{\ell i} > 0$ for $\ell \neq r$. If $c_{ri} > 0$, we have $\nu_{ri} = 0$ from (6.43). Thus, we have

$$c_{ri} = \left(\sqrt{\frac{\sigma_i \lambda_i}{\beta_r}} - \lambda_i \right)^+ . \tag{6.47}$$

Note that $c_{ri} > 0$ requires $\frac{\sigma_i}{\lambda_i} > \beta_r$, i.e., $\frac{1}{\lambda_i} \frac{u_i}{u_i+\lambda_i} \prod_{\ell \neq r} \frac{c_{\ell i}}{c_{\ell i}+\lambda_i} > \beta_r$ which also implies that if $\frac{1}{\lambda_i} \frac{u_i}{u_i+\lambda_i} \prod_{\ell \neq r} \frac{c_{\ell i}}{c_{\ell i}+\lambda_i} \leq \beta_r$, then we must have $c_{ri} = 0$. We repeat this step for $r = 1, \ldots, m$.

Next, we solve for u_is for given c_{ri}s for all r with $c_{ri} > 0$. We rewrite (6.40) as

$$(u_i + \lambda_i)^2 = \frac{\rho_i \lambda_i}{\theta - \eta_i}, \tag{6.48}$$

where $\rho_i = \prod_{r=1}^{m} \frac{c_{ri}}{c_{ri}+\lambda_i}$. Then, we find u_i as

$$u_i = \sqrt{\frac{\rho_i \lambda_i}{\theta - \eta_i}} - \lambda_i, \tag{6.49}$$

for all i with $c_{ri} > 0$ for all r. If $u_i > 0$, we have $\eta_i = 0$ from (6.44). Thus, we have

$$u_i = \left(\sqrt{\frac{\rho_i \lambda_i}{\theta}} - \lambda_i \right)^+. \tag{6.50}$$

Similarly, $u_i > 0$ requires $\frac{\rho_i}{\lambda_i} > \theta$, i.e., $\frac{1}{\lambda_i}\prod_{r=1}^{m} \frac{c_{ri}}{c_{ri}+\lambda_i} > \theta$ which implies that if $\frac{1}{\lambda_i}\prod_{r=1}^{m} \frac{c_{ri}}{c_{ri}+\lambda_i} \leq \theta$, then we must have $u_i = 0$. Thus, similar to the results in Section 6.5, for given c_{ri}s (resp. for given u_is and $c_{\ell i}$s for all $\ell \neq r$), we observe that the optimal rate allocation policy for the user (resp. for cache r) is a threshold policy and the update rates for the user (resp. for cache r) are equal to zero for the files with very large λ_is.

Similar to Lemma 6.1, one can show that in the optimal policy, if $c_{ri} = 0$ for some r, then we must have $c_{\ell i} = 0$ for all $\ell \neq r$ and $u_i = 0$. Furthermore, if $u_i = 0$ for some i, then we must have $c_{ri} = 0$ for all r. One can also show that the total update rate constraints for cache r, i.e., $\sum_{i=1}^{n} c_{ri} \leq C_r$, for all r and for the user, i.e., $\sum_{i=1}^{n} u_i \leq U$, should be satisfied with equality as the objective function in (6.37) is an increasing function of c_{ri} and u_i. Thus, in the optimal policy, we must have $\sum_{i=1}^{n} c_{ri} = C_r$ for all r and $\sum_{i=1}^{n} u_i = U$.

We note from (6.47) that the update rates of cache r directly depend on the update rates of the other caches as well as the update rates of the user. Similarly, we note from (6.50) that the update rates of the user directly depend on the update rates of all caches. In order to find the overall solution, for given initial u_is and $c_{\ell i}$ for all $\ell \neq r$, we choose $c_{ri} = 0$ for the files with $u_i = 0$ or $c_{\ell i} = 0$ for some ℓ. We solve $\sum_{i=1}^{n} c_{ri} = C_r$ for β_r and then, find c_{ri}s by using (6.47) similar to the solution method in Section 6.5. We repeat this step for $r = 1, \dots, m$. Next, for given c_{ri}s, we find u_is by following a similar procedure. We keep updating these parameters until $(c_{1i}, \dots, c_{ri}, u_i)$ tuples converge.

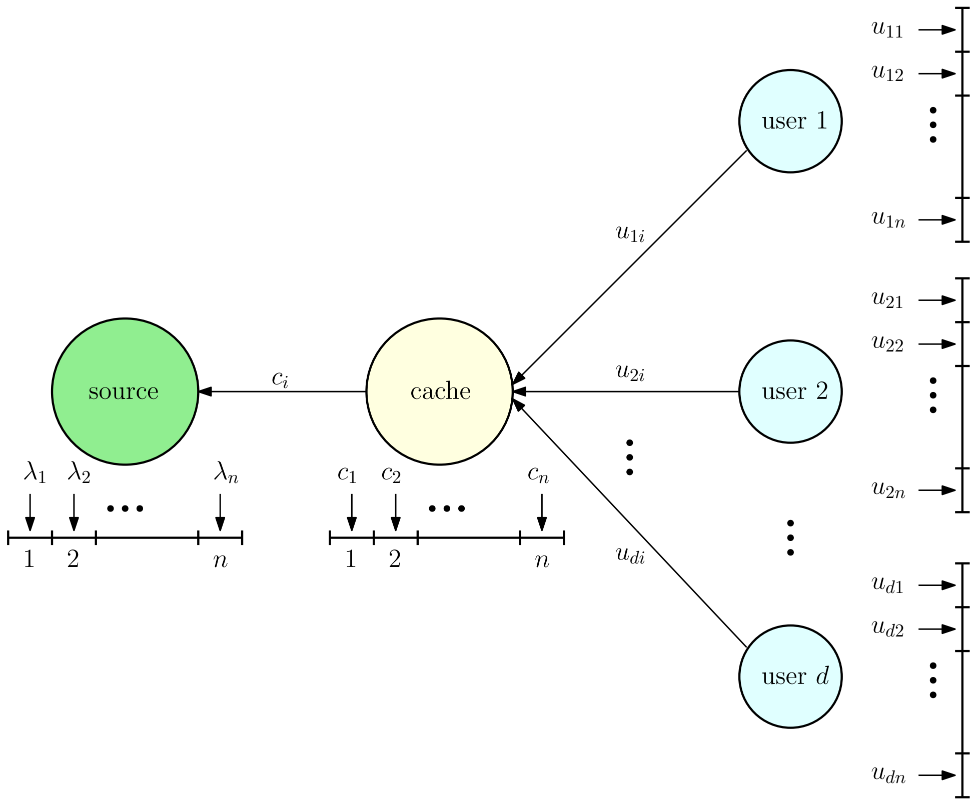
In the following section, we find rate allocations for a system with a source, a single cache, and multiple users.

6.7 Freshness Maximization for a System with Multiple Users

In this section, we consider a system where there is a source, a single cache and d users connected to the cache, as shown in Fig. 6.6. Our aim is to find the update rates for the cache and for the users such that the overall freshness experienced by the users is maximized.

Let the kth user's update rate for the ith file be u_{ki}. Each user is subject to a total update rate constraint as $\sum_{i=1}^{n} u_{ki} \leq U_k$, for $k = 1, \dots, d$, and the cache is subject to a total update rate constraint as $\sum_{i=1}^{n} c_i \leq C$. From Section 6.3, the average freshness of the ith file at the kth user is $F_u(k,i) = \sum_{i=1}^{n} \frac{u_{ki}}{u_{ki}+\lambda_i} \frac{c_i}{c_i+\lambda_i}$. Then, we write the freshness maximization problem as

$$\begin{aligned} \max_{\{c_i, u_{ki}\}} \quad & \sum_{k=1}^{d} \sum_{i=1}^{n} \frac{u_{ki}}{u_{ki}+\lambda_i} \frac{c_i}{c_i+\lambda_i} \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i \leq C \\ & \sum_{i=1}^{n} u_{ki} \leq U_k, \quad k = 1, \dots, d \\ & c_i \geq 0, \quad u_{ki} \geq 0, \quad k = 1, \dots, d, \quad i = 1, \dots, n. \end{aligned} \tag{6.51}$$



source
cache
user 1
user 2
user d
c_i
u_{1i}
u_{2i}
u_{di}
λ_1 λ_2 λ_n
1 2 n
c_1 c_2 c_n
1 2 n
u_{11} u_{12} u_{1n}
u_{21} u_{22} u_{2n}
u_{d1} u_{d2} u_{dn}

Figure 6.6: A cache updating system with a source, a single cache and d users.

We introduce the Lagrangian function for (6.51) as

$$\begin{aligned}\mathcal{L} =& -\sum_{k=1}^{d}\sum_{i=1}^{n}\frac{u_{ki}}{u_{ki}+\lambda_i}\frac{c_i}{c_i+\lambda_i} + \beta\left(\sum_{i=1}^{n}c_i - C\right) + \sum_{k=1}^{d}\theta_k\left(\sum_{i=1}^{n}u_{ki} - U_k\right)\\ &-\sum_{i=1}^{n}\nu_i c_i - \sum_{k=1}^{d}\sum_{i=1}^{n}\eta_{ki}u_{ki},\end{aligned} \tag{6.52}$$

where $\beta \geq 0$, $\theta_k \geq 0$, $\nu_i \geq 0$ and $\eta_{ki} \geq 0$. We write the KKT conditions as

$$\frac{\partial \mathcal{L}}{\partial c_i} = -\frac{\lambda_i}{(c_i+\lambda_i)^2}\sum_{k=1}^{d}\frac{u_{ki}}{u_{ki}+\lambda_i} + \beta - \nu_i = 0, \tag{6.53}$$

$$\frac{\partial \mathcal{L}}{\partial u_{ki}} = -\frac{\lambda_i}{(u_{ki}+\lambda_i)^2}\frac{c_i}{c_i+\lambda_i} + \theta_k - \eta_{ki} = 0, \tag{6.54}$$

for all k and i. The complementary slackness conditions are

$$\beta\left(\sum_{i=1}^{n} c_i - C\right) = 0, \tag{6.55}$$

$$\theta_k\left(\sum_{i=1}^{n} u_{ki} - U_k\right) = 0, \tag{6.56}$$

$$\nu_i c_i = 0, \tag{6.57}$$

$$\eta_{ki} u_{ki} = 0. \tag{6.58}$$

The objective function in (6.51) is not jointly concave in c_i and u_{ki}. However, for given u_{ki}s, the objective function in (6.51) is concave in c_i, and for given c_is, the objective function in (6.51) is jointly concave in all u_{ki}. Thus, similar to the solution approach used in Section 6.5, we apply an alternating maximization based method to find (c_i, u_{ki}) such that (6.53) and (6.54) are satisfied for all k and i.

For given u_{ki}s, we find c_i as

$$c_i = \frac{1}{\sqrt{\beta - \nu_i}}\sqrt{\sum_{k=1}^{d} \frac{u_{ki}\lambda_i}{u_{ki} + \lambda_i}} - \lambda_i, \tag{6.59}$$

for all i. If $c_i > 0$, we have $\nu_i = 0$ due to (6.57). Thus, we have

$$c_i = \left(\frac{1}{\sqrt{\beta}}\sqrt{\sum_{k=1}^{d} \frac{u_{ki}\lambda_i}{u_{ki} + \lambda_i}} - \lambda_i\right)^{+}. \tag{6.60}$$

Similarly, the optimal rate allocation policy is a threshold based policy where if $\frac{1}{\lambda_i}\left(\sum_{k=1}^{d} \frac{u_{ki}}{u_{ki}+\lambda_i}\right) < \beta$, then we have $c_i = 0$.

Next, for a given c_i with $c_i > 0$, we find u_{ki} as

$$u_{ki} = \frac{1}{\sqrt{\theta_k - \eta_{ki}}} \sqrt{\frac{c_i \lambda_i}{c_i + \lambda_i}} - \lambda_i, \tag{6.61}$$

for all i. If $u_{ki} > 0$, we have $\eta_{ki} = 0$ due to (6.58). Thus, we have

$$u_{ki} = \left(\frac{1}{\sqrt{\theta_k}} \sqrt{\frac{c_i \lambda_i}{c_i + \lambda_i}} - \lambda_i \right)^+ . \tag{6.62}$$

Thus, we note that the optimal rate allocation policy is a threshold policy where if $\frac{1}{\lambda_i} \frac{c_i}{c_i + \lambda_i} < \theta_k$, then we have $u_{ki} = 0$. We observe that the optimal rates for the users depend directly on λ_i and c_i. As the update rates for the cache depend on the update rates of all the users, update rates of the users affect each other indirectly. We also note that in Section 6.6, where the caches are connected serially, we see this effect as a product of the terms, i.e., $\prod_{r=1}^{m} \frac{c_{ri}}{c_{ri} + \lambda_i}$. In this section, as the users are connected in parallel to a single cache, we see this effect as a summation of the terms, i.e., $\sum_{k=1}^{d} \frac{u_{ki}}{u_{ki} + \lambda_i}$.

In the next section, we provide numerical results for the system with a single cache in Section 6.5, with multiple caches in Section 6.6, and with multiple users in Section 6.7.

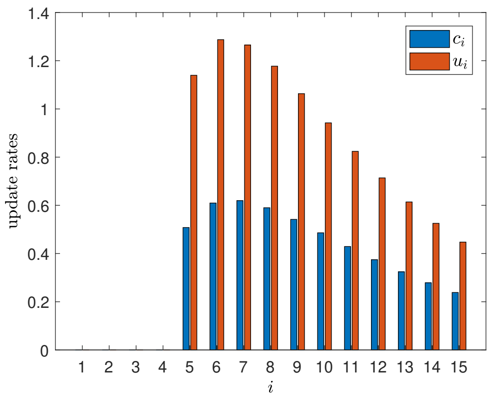
6.8 Numerical Results

In this section, we provide five numerical results. For these results, we use the following update arrival rates at the source

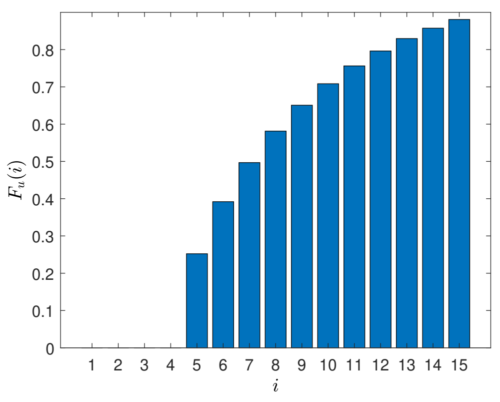
$$\lambda_i = bq^i, \quad i = 1, \dots, n, \tag{6.63}$$

where $b > 0$ and $0 < q \leq 1$ such that $\sum_{i=1}^{n} \lambda_i = a$. Note that with the update arrival rates at the source in (6.63), we have $\lambda_i \geq \lambda_j$ for $i \leq j$.

In example 1, we take $a = 10$, $q = 0.7$, and $n = 15$ in (6.63). For this example, we consider the system with a source, a single cache and a user. We choose the total update rate constraint for the cache as $C = 5$, i.e., $\sum_{i=1}^{n} c_i \leq 5$, and for the user as $U = 10$, i.e., $\sum_{i=1}^{n} u_i \leq 10$. We initialize the file update rates at the user as $u_i = \frac{U}{n}$ for all i. We apply the alternating maximization method in Section 6.5 to find the update rates for the cache and for the user until the KKT conditions in (6.23) and (6.24) are satisfied. We see in Fig. 6.7(a) that the first four files which are updated most frequently at the source are not updated by the cache and the user. As these files change too frequently at the source, their stored versions at the user become obsolete very quickly. In other words, updating files that change less frequently at the source brings more contribution to the overall information freshness at the user. The distributions of the update rates for the cache and for the user have similar trends as the update rates increase up to the seventh file for the cache and the sixth file for the user, and then update rates for the cache and the user decrease. Even



(a)



(b)

Figure 6.7: (a) Update rate allocation for the cache and the user for each file, and (b) the corresponding freshness $F_u(i)$, when $U = 10$ and $C = 5$ with the file update rates at the source λ_i given in (6.63), with $a = 10$ and $q = 0.7$ for $n = 15$.

though the update rates of the cache and the user for the slowly changing files are low, we see in Fig. 6.7(b) that the freshness of the slowly varying files is higher compared to the rapidly changing files.

In example 2, we consider the same system as in example 1, and compare the performance of the proposed updating policy which maximizes the freshness at the user with two other baseline updating policies. As the first baseline policy, we consider λ-*proportional* update policy where the update rates for the cache and for the user are chosen as $c_i = \frac{\lambda_i C}{\sum_{i=1}^n \lambda_i}$ and $u_i = \frac{\lambda_i U}{\sum_{i=1}^n \lambda_i}$ for all i, respectively. With this policy, the freshness of the ith file at the user is $F_u(i) = \frac{U}{U+a}\frac{C}{C+a}$. Thus, this update policy may be desirable if the user wants to have the same level of freshness for all files. As the second baseline policy, we consider λ-*inverse* update policy where the update rates for the cache and for the user are given by $c_i = C\frac{\frac{1}{\lambda_i}}{\sum_{i=1}^n \frac{1}{\lambda_i}}$ and $u_i = U\frac{\frac{1}{\lambda_i}}{\sum_{i=1}^n \frac{1}{\lambda_i}}$ for all i, respectively. The motivation for λ-inverse update policy stems from the fact that updating slowly varying files at the source with higher update rates brings more contribution to the overall freshness. In this example, we take $C = 15$, $U = 10$, and λ_i in (6.63) with $n = 20$. In Fig. 6.8(a), we fix the total file update rate at the source, i.e., $a = 10$, and vary the distribution of the file update rate at the source by choosing $0 < q \leq 1$. We observe in Fig. 6.8(a) that proposed update policy achieves the highest freshness compared to the considered baseline policies. We see that the λ-proportional policy achieves a constant level of freshness independent of the distribution of the file change rate at the source, but this policy gives the lowest freshness compared to others. We note that when λ_is are evenly distributed, i.e., when q is close to 1, we see that all three updating policies achieve similar levels of freshness. In Fig. 6.8(b), we fix $q = 0.7$ and increase the file change rate at the source, i.e., $a = 1, \dots, 20$. We observe in Fig. 6.8(b) that the proposed update policy achieves the highest freshness. When the total file change rate at the source is low, λ-proportional update policy achieves freshness close to the proposed update policy, whereas when the total file change rate at the source is high, λ-inverse achieves freshness close to the proposed update policy.

In example 3, we consider the same system as in examples 1 and 2 but this time we examine the effect of file update rates at the source over the information freshness at the user. We take λ_i in (6.63) with $a = 2$, $n = 15$ for $q = 0.5, 0.75, 1$. We note that a smaller q corresponds to a less even (more polarized) distribution of file change rates at the source. We choose the total update rate for the user as $U = 10$ and vary the total update rate for the cache as $C = 1, 1.5, \dots, 10$. For each C and q values, we initialize $u_i = \frac{U}{n}$ for all i and apply the alternating maximization method proposed

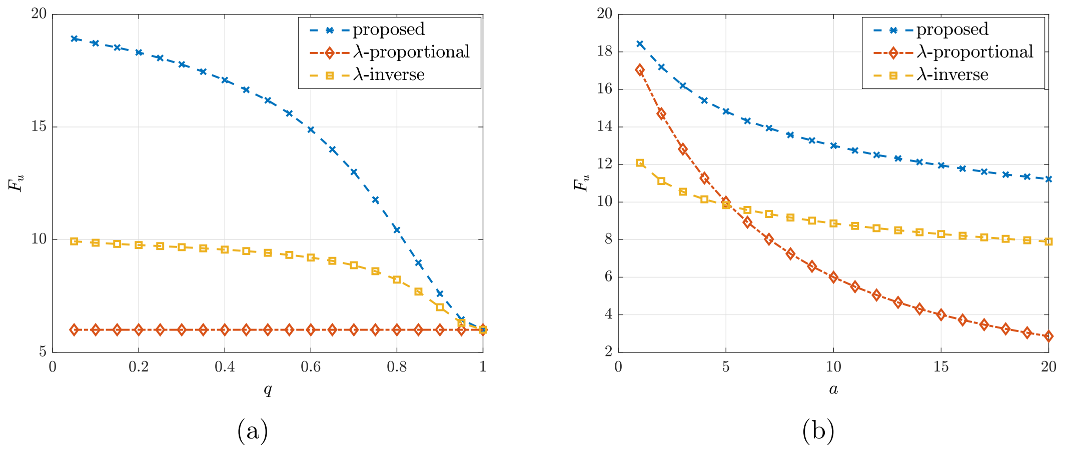


Figure 6.8: We compare the proposed update policy with the λ-proportional and the λ-inverse updating policies when $C = 15$, $U = 10$. We use λ_i in (6.63) for $n = 20$, (a) $a = 10$, and $0 < q \leq 1$ and (b) $q = 0.7$ and $a = 1, \ldots, 20$.

in Section 6.5 until convergence. We see in Fig. 6.9 that when the distribution of the change rates of the files are more polarized, i.e., when q is small, the overall information freshness at the user is larger as the freshness contribution from the slowly varying files can be utilized more with the more polarized distributions of file change rates at the source. We also note that for a fixed λ_i distribution, the freshness of the user increases with the total update rate at the cache C due to the fact that the user gets fresh files more frequently from the cache as the freshness of the files at the cache increases with C.

In example 4, we consider a system where there are two caches placed in between the source and the user with λ_i as given in (6.63) with $a = 10$, $q = 0.7$, and $n = 10$. We take the total update rate constraint for the user as $U = 20$ and, for the second cache as $C_2 = 10$. We initialize the update rates at the user and at the second cache as $u_i = \frac{U}{n}$ and $c_{2i} = \frac{C_2}{n}$ for all i, respectively, and apply the

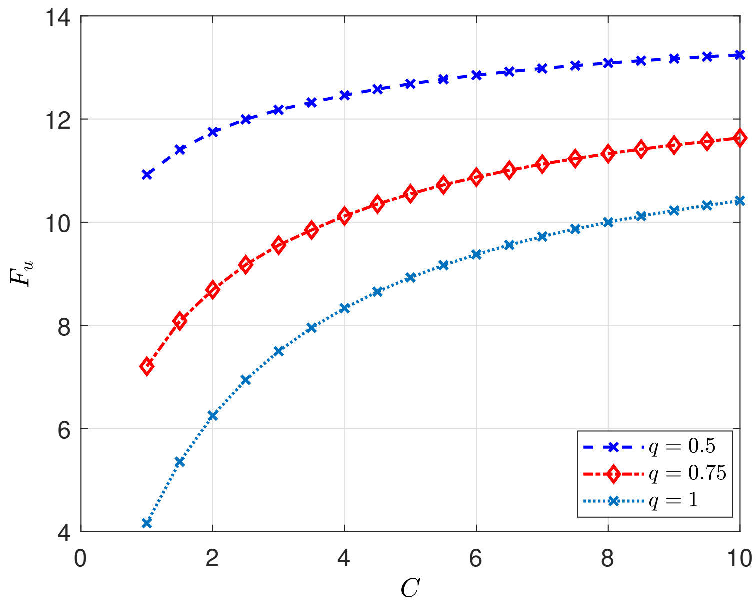
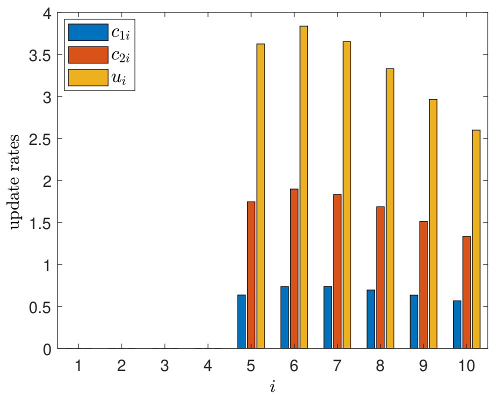
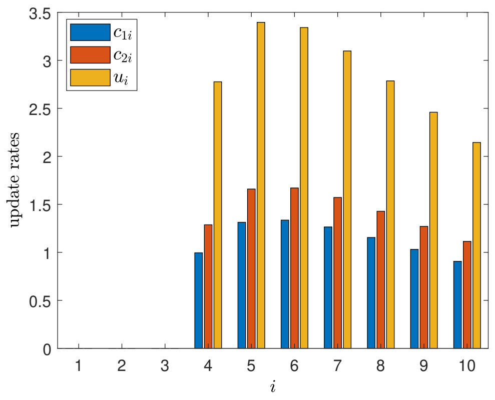


Figure 6.9: Total freshness of the user F_u with respect to C, when λ_i are given in (6.63), with $a = 2$ and $q = 0.5, 0.75, 1$ for $n = 15$.

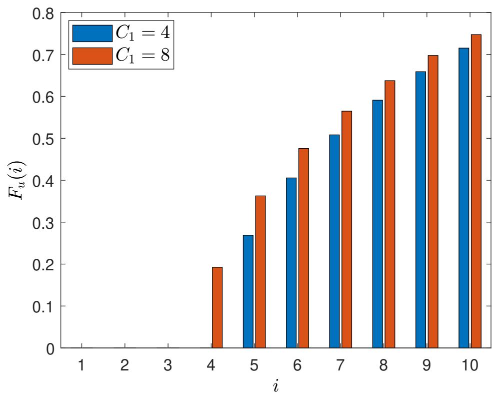
alternating maximization method proposed in Section 6.6, for the total update rate constraint for the first cache as $C_1 = 4, 8$. We observe that the update rates for the caches and for the user have similar trends as shown in Fig. 6.10(a) for $C_1 = 4$ and in Fig. 6.10(b) for $C_1 = 8$. We note that the update rates of the caches in (6.47) depend directly on the update rates of the other caches and also of the user. Similarly, the update rates of the user in (6.50) depend directly the update rates of all caches. That is why even though the total update rate constraints for the second cache and for the user remain the same in Fig. 6.10(a)-(b), we see that the update rates at the second cache and at the user also change depending on the update rates at the first cache. In Fig. 6.10(c), we observe that increasing the total update rate constraint for the first cache improves the freshness of every file except the first three files as the total update rate constraints for the caches and for the user are not high



(a)



(b)

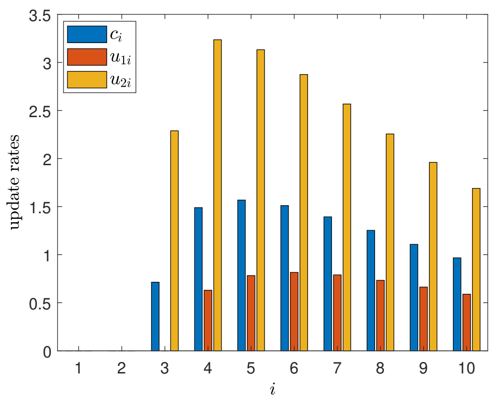


(c)

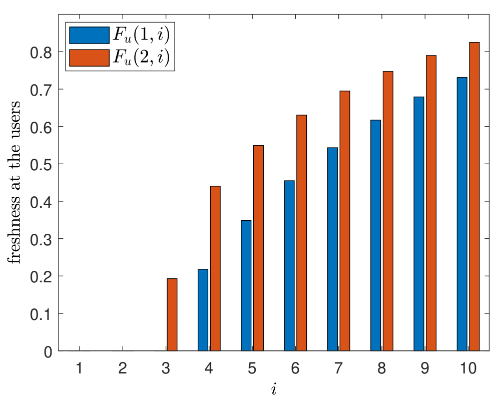
Figure 6.10: The update rates of the caches and the user when the total update rate constraint for the user is $U = 20$ and, for the second cache is $C_2 = 10$. Total update rate constraint for the first cache is (a) $C_1 = 4$, and (b) $C_1 = 8$. The freshness of the files at the user is shown in (c).

enough to update the most rapidly changing files. Furthermore, the improvement on the freshness of the rapidly changing files is more significant than the others as the freshness of the files at the user is a concave increasing function of c_{1i}.

In example 5, we consider the system where there is a source, a cache and two users connected to the cache. For this example, we use λ_i in (6.63) with $a = 10$, $q = 0.7$ and $n = 10$. We take the total update rate constraint for the cache as



(a)



(b)

Figure 6.11: A system with a source, a single cache and two users: (a) The update rates of the cache and of the users, (b) the freshness of each file at the users.

$C = 10$, for the first user as $U_1 = 5$ and for the second user as $U_2 = 20$. We initialize the cache update rates as $c_i = \frac{C}{n}$ for all i, and apply the proposed alternating maximization based method to find (c_i, u_{1i}, u_{2i}) that satisfy the KKT conditions in (6.53) and (6.54). The update rates of the cache and the users are shown in Fig. 6.11(a) where we see that even though the update rate of the third file at the first user is equal to zero, the cache still updates the third file for the second user. We see in Fig. 6.11(b) that the freshness of the files at the second user is higher as the total update rate of the second user is higher compared to the first user. The freshness difference between the users decreases for the slowly changing files at the source.

6.9 Conclusion

In this chapter, we first considered a cache updating system with a source, a single cache and a user. We found an analytical expression for the average freshness of the files at the cache and also at the user. We generalized this setting to the case where there are multiple caches placed in between the source and the user. Then, we provided an alternating maximization based method to find the update rates for the cache(s) and for the user to maximize the total freshness of the files at the user. We observed that for a given set of update rates of the user (resp. of the cache), the optimal rate allocation policy for the cache (resp. for the user) is a threshold policy where the frequently changing files at the source may not be updated by the user and also by the cache. Finally, we considered a system with multiple users refreshed via a single cache and found update rates for the cache and the users to maximize the overall freshness of the users.

CHAPTER 7

Freshness in Cache Updating Systems with Limited Storage Capacity

7.1 Introduction

In this chapter, we consider a cache updating system that consists of a source, a cache with limited cache (i.e., storage) capacity and a user as shown in Fig. 7.1. In this system, the source keeps the freshest versions of all the files that are refreshed with known rates λ_i. The cache gets the freshest versions of the files from the source, but its cache capacity is limited, i.e., it can only store the freshest versions of K files where $K \leq n$. The user gets files either from the cache or from the source. If the user gets a file from the cache, the updated file at the user might still be outdated depending on the file status at the source. If the user gets a file directly from the source, the received file is always fresh. However, as the channel between the user and the source is not perfect, there is a file transmission time which decreases the freshness at the user. Thus, in this chapter, we study the trade-off between storing the files at the cache to decrease the file transmission times versus directly obtaining the fresh files from the source at the expense of higher transmission times. Our aim

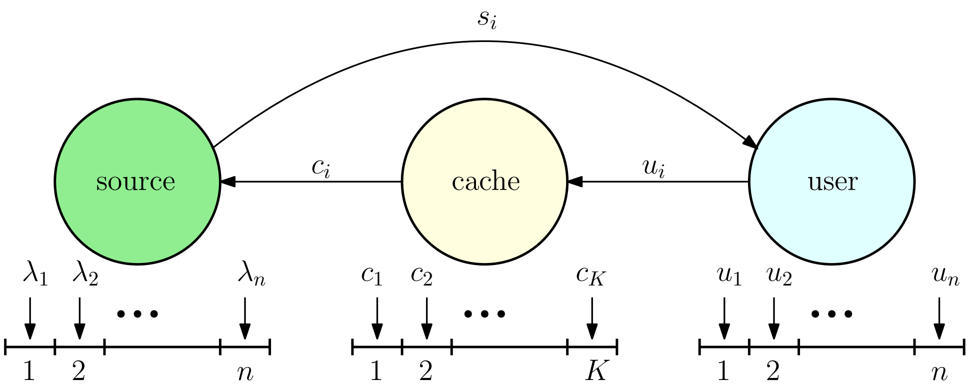


Figure 7.1: A cache updating system with a source, a cache and a user.

is to find the optimal caching status for each file (i.e., whether to store the file at the cache or not) and the corresponding optimal file update rates at the cache.

References that are most closely related to our work are [93] and [101]. Reference [93] considers a model where a resource constrained remote server wants to keep the items at a local cache as fresh as possible. Reference [93] shows that the update rates of the files should be chosen proportional to the square roots of their popularity indices. Different from [93] where the freshness of the local cache is considered, we consider the freshness at the end-user. Furthermore, the freshness metric that we use in this chapter is different than the traditional age metric used in [93], and hence our overall work is distinct compared to [93]. In comparison to our earlier work in [101] (also Chapter 6 in this thesis), here, we consider a cache with limited caching capacity, and we study the trade-off between storing the files at the cache and obtaining the files directly from the source.

In this chapter, we find an analytical expression for the average freshness of the files at the user when the files are cached and not cached. We impose a total update rate constraint for the cache due to limited nature of resources. We find the optimal caching status for each file and the corresponding optimal file update rates at the cache. We observe that due to binary nature of file caching status, the optimization problem is NP-hard. However, for a given set of caching status of the files, the problem becomes a convex optimization problem in terms of the file update rates at the cache. For a given set of caching status of the files, the optimal rate allocation policy at the cache is a *threshold policy* where the rapidly changing files at the source may not be updated. We observe that when the total update rate of the cache is high, storing files at the cache improves the freshness of the user. However, when the total update rate of the cache is low, it is optimal for the user to obtain the rapidly changing files and the files that have relatively small transmission times directly from the source.

7.2 System Model

We consider an information updating system where there is a source, a cache and a user as shown in Fig. 7.1. The source keeps the freshest version of n files which are updated with exponential inter-arrival times with rate λ_i. The file updates at the source are independent of each other. The cache gets fresh files from the source, but it may store only K files where $K = 1, \dots, n$. We assume that the channel between the source and the cache is perfect and the transmission times are negligible. Thus, if the cache requests an update for a stored file, it receives the file from the source right away. We model the inter-update request times for the ith file at the cache as exponential with rate c_i. The cache is subject to a total update rate constraint, i.e.,

$\sum_{i=1}^{n} c_i \leq C$ as in [93, 101].

The inter-update request times of the user for the ith file are exponential with rate u_i. The channel between the user and the cache is also assumed to be perfect and the transmission times are negligible. Thus, if the requested file is stored at the cache, the user gets the stored file at the cache right away. If the user requests a file which is not cached, the cache forwards the file update request from the user to the source. Since the cache only forwards the user requests for uncached files (i.e., without creating requests of its own), $c_i > u_i$ is not possible, and we have $c_i \leq u_i$. For uncached files, the file update requests at the cache are fully synchronized with the user's requests which means that when the user requests an update for an uncached file, this request reaches the source immediately if the cache forwards it. Thus, for each file update request of the user for the uncached file i, the cache forwards the request to the source with probability $p_i = \frac{c_i}{u_i}$. From [147, Thm. 13.6], the effective inter-update request times of the user for an uncached file are exponential with rates c_i. We assume that the channel between the source and the user is imperfect and the transmission time for the ith file is exponential with rate s_i.

We note that each file at the source is always *fresh*. However, when a file is updated at the source, the stored versions of the same file at the cache and at the user become *outdated*. When the cache gets an update for an outdated file, the updated file in the cache becomes *fresh* again until the next update arrival at the source. The user gets files either from the cache or from the source. If the user gets a file from the cache, it will receive the file update immediately, but the received file can be outdated if the file at the cache is not fresh. If the user gets a file directly

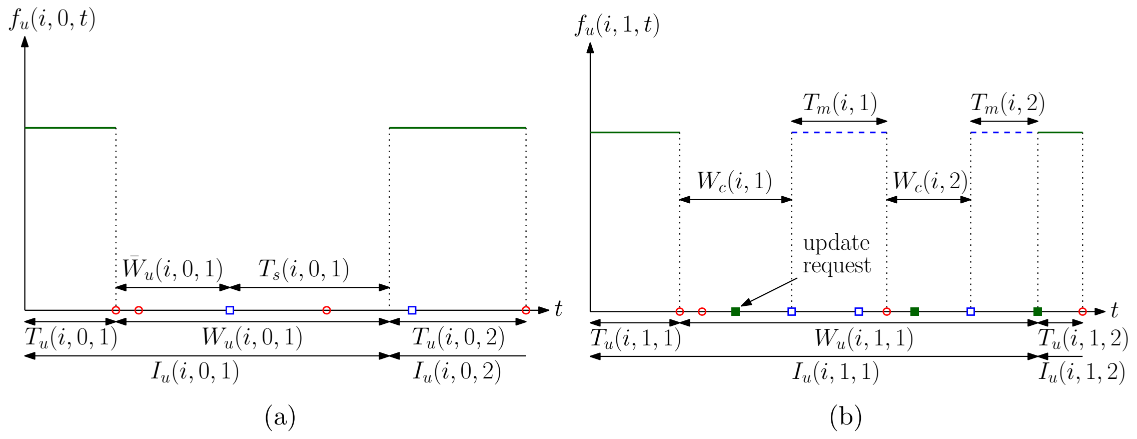


Figure 7.2: Sample evolution of the freshness of the ith file at the user when the ith file is (a) not cached and (b) cached. Red circles represent the update arrivals at the source, blue squares represent the update requests from the cache, and green filled squares represent the update requests from the user.

from the source, the received file is always fresh, but the transmission takes time. We note that since the cache and the user are unaware of the file updates at the source, they do not know whether they have the freshest versions of the files or not. Thus, they may still unknowingly request an update even though they have the freshest version of a file.

We use k_i which is a binary variable to indicate the caching status of the ith file, i.e., $k_i = 1$ when the ith file is cached and $k_i = 0$ when it is not cached. We define $f_u(i, k_i, t)$ as the freshness function of the ith file at the user as,

$$f_u(i,k_i,t) = \begin{cases} 1, & \text{if the } i\text{th file is fresh at time } t, \\ 0, & \text{otherwise,} \end{cases} \tag{7.1}$$

where the instantaneous freshness function is a binary function taking values of fresh, "1", or not fresh, "0", at any time t. A sample $f_u(i, k_i, t)$ is shown in Fig. 7.2(a) when $k_i = 0$ and in Fig. 7.2(b) when $k_i = 1$.

File updates that replace an outdated version of the file with the freshest one are denoted as *successful* updates. We define the time interval between the jth and the $(j+1)$th successful updates for the ith file at the user as the jth update cycle and denote it by $I_u(i, k_i, j)$. We denote the time duration when the ith file at the user is fresh during the jth update cycle as $T_u(i, k_i, j)$. We denote $f_c(i, t)$ as the freshness function of the ith file at the cache. Similarly, update cycles and duration of freshness at the cache are denoted by $I_c(i, j)$ and $T_c(i, j)$. Then, we denote $F_u(i, 1)$ (resp. $F_u(i, 0)$) as the long term average freshness of the ith file at the user when the file is cached (resp. when the file is not cached), i.e., $k_i = 1$ (resp. $k_i = 0$). $F_u(i, k_i)$ is equal to

$$F_u(i, k_i) = \lim_{T \to \infty} \frac{1}{T} \int_0^T f_u(i, k_i, t) dt. \tag{7.2}$$

Similar to [1], we have

$$F_u(i, k_i) = \lim_{T \to \infty} \frac{N}{T} \left(\frac{1}{N} \sum_{j=1}^{N} T_u(i, k_i, j) \right) = \frac{\mathbb{E}[T_u(i, k_i)]}{\mathbb{E}[I_u(i, k_i)]},$$

where N is the number of update cycles in time duration T. We define the total freshness over all files at the user F_u as

$$F_u = \sum_{i=1}^{n} k_i F_u(i, 1) + (1 - k_i) F_u(i, 0). \tag{7.3}$$

Our aim is to find the optimal file caching status k_i, and the corresponding file update rates at the cache c_i for $i = 1, \dots, n$, such that the total average freshness of the user F_u is maximized while satisfying the constraints on the cache capacity, i.e., $\sum_{i=1}^n k_i \leq K$, the total update rate of the cache, $\sum_{i=1}^n c_i \leq C$, and the feasibility constraints i.e., $c_i \leq u_i$ for uncached files (for files with $k_i = 0$). Thus, our problem is,

$$\begin{aligned}
\max_{\{k_i, c_i\}} \quad & F_u \\
\text{s.t.} \quad & \sum_{i=1}^n k_i \leq K \\
& \sum_{i=1}^n c_i \leq C \\
& (1-k_i)c_i \leq u_i, \qquad i = 1, \dots, n \\
& c_i \geq 0, \quad k_i \in \{0,1\}, \quad i = 1, \dots, n.
\end{aligned} \tag{7.4}$$

In the following section, we find the long term average freshness of the ith file at the user $F_u(i, k_i)$ when the ith file is cached and when it is not cached. Once we find $F_u(i, k_i)$, this will determine the objective function of (7.4) via (7.3).

7.3 Average Freshness Analysis

In this section, we find the long term average freshness for the ith file at the user $F_u(i, k_i)$ for $k_i \in \{0, 1\}$. In the following theorem, we first find the long term average freshness of the ith file at the user when the ith file is cached.

Theorem 7.1 *If the ith file is cached, the long term average freshness of the ith file* *at the user $F_u(i,1)$ is equal to*

$$F_u(i,1) = \frac{\mathbb{E}[T_u(i,1)]}{\mathbb{E}[I_u(i,1)]} = \frac{u_i}{u_i+\lambda_i}\frac{c_i}{c_i+\lambda_i}. \tag{7.5}$$

The proof of the Theorem 7.1 follows from [101, Section III]. Since the user gets fresh files more frequently from the cache for higher values of c_i, the freshness of the ith file at the user $F_u(i,1)$ in (7.5) increases with c_i. In addition, $F_u(i,1)$ in (7.5) is a concave function of c_i. If the user was directly connected to the source, freshness of the ith file at the user would be equal to $\frac{u_i}{u_i+\lambda_i}$ as in [101]. However, as the user is connected to the source via the cache, the freshness experienced by the user proportionally decreases with the freshness of the cache which is $\frac{c_i}{c_i+\lambda_i}$. Note that $\frac{c_i}{c_i+\lambda_i} < 1$ for all c_i.

Next, we find the long term average freshness of the ith file at the user $F_u(i,0)$ when the ith file is not cached.

Theorem 7.2 *If the ith file is not cached, the long term average freshness of the ith file at the user $F_u(i,0)$ is equal to*

$$F_u(i,0) = \frac{\mathbb{E}[T_u(i,0)]}{\mathbb{E}[I_u(i,0)]} = \frac{c_i}{c_i+\lambda_i+\frac{c_i\lambda_i}{s_i}}. \tag{7.6}$$

Proof: When the ith file at the user becomes fresh, the time until the next file update arrival at the source is still exponentially distributed with rate λ_i due to the memoryless property of the exponential distribution. Thus, $\mathbb{E}[T_u(i,0)] = \frac{1}{\lambda_i}$.

After the ith file is updated at the source, the stored version of the ith file at the user becomes outdated, i.e., the instantaneous freshness function $f_u(i,0,t)$ becomes 0 again. We denote the time interval until the source gets a file update request for the ith file after the file at the user becomes outdated as $\bar{W}_u(i,0)$ which is exponentially distributed with rate c_i as discussed in Section 7.2. After receiving the file update request from the user, the source sends the ith file directly to the user. If the ith file at the source is updated during a file transfer, then the file transfer is interrupted and the fresh file is sent until the freshest version of the ith file is successfully transmitted to the user. We denote the total transmission time for the ith file as $T_s(i,0)$. Due to [147, Prob. 9.4.1], $T_s(i,0)$ is also exponentially distributed with rate s_i. Thus, we have $\mathbb{E}[T_s(i,0)] = \frac{1}{s_i}$. We denote the time interval when the ith file at the user is outdated during the jth update cycle as $W_u(i,0,j)$, i.e., $W_u(i,0,j) = I_u(i,0,j) - T_u(i,0,j)$, which is also equal to $W_u(i,0,j) = \bar{W}_u(i,0,j) + T_s(i,0,j)$. We denote the typical random variables for $W_u(i,0,j)$ and $I_u(i,0,j)$ as $W_u(i,0)$ and $I_u(i,0)$, respectively. Then, we have $\mathbb{E}[W_u(i,0)] = \mathbb{E}[\bar{W}_u(i,0)] + \mathbb{E}[T_s(i,0)] = \frac{1}{c_i} + \frac{1}{s_i}$ and

$$\mathbb{E}[I_u(i,0)] = \mathbb{E}[T_u(i,0)] + \mathbb{E}[W_u(i,0)] = \frac{1}{\lambda_i} + \frac{1}{c_i} + \frac{1}{s_i}.$$

Thus, we get $F_u(i,0)$ in (7.6) by using $F_u(i,0) = \frac{\mathbb{E}[T_u(i,0)]}{\mathbb{E}[I_u(i,0)]}$. ■

We note that $F_u(i,0)$ in (7.6) is an increasing function of c_i and also is concave in c_i. When the user gets a file from the source directly, the received file is always fresh, but due to the transmission time between the source and the user, the average time that the ith file is outdated at the user increases. Thus, the freshness of the

ith file at the user $F_u(i,0)$ in (7.6) increases with s_i. Further, $F_u(i,1) > F_u(i,0)$ implies that $\frac{c_i}{c_i+\lambda_i} > \frac{s_i}{u_i}$. In other words, if the file update rate of the ith file at the cache c_i is high enough, it is better to cache file i. However, if file i is updated too frequently at the source, i.e., λ_i is too large, or file i has small transmission times, i.e., s_i is too high, then it is better to get the file from the source. Thus, there is a trade-off: If a file is stored at the cache, this enables the user to obtain the file more quickly, but the received file might be outdated. On the other hand, if the user gets the file directly from the source, the file will always be fresh, but the file transmission time decreases the freshness at the user.

7.4 Freshness Maximization

In this section, we solve the optimization problem in (7.4). Using $F_u(i,k_i)$ in (7.5) and (7.6) and F_u in (7.3), we rewrite the freshness maximization problem in (7.4) as

$$\begin{aligned}
\max_{\{k_i,c_i\}} \quad & \sum_{i=1}^{n} k_i \frac{u_i}{u_i+\lambda_i}\frac{c_i}{c_i+\lambda_i} + (1-k_i)\frac{c_i}{c_i+\lambda_i+\frac{c_i\lambda_i}{s_i}} \\
\text{s.t.} \quad & \sum_{i=1}^{n} k_i \leq K \\
& \sum_{i=1}^{n} c_i \leq C \\
& (1-k_i)c_i \leq u_i, \qquad i=1,\dots,n \\
& c_i \geq 0, \quad k_i \in \{0,1\}, \quad i=1,\dots,n.
\end{aligned} \tag{7.7}$$

In order to solve the optimization problem in (7.7), we need to determine the optimal caching status for each file k_i and find the optimal file update rates at the cache c_i. We note that the optimization problem in (7.7) is NP-hard due to the presence of binary variables k_i. However, for a given $(k_1, k_2, \dots, k_n)$ tuple, (7.7) becomes a convex optimization problem in c_i. Thus, the optimal solution can be found by searching over all possible $(k_1, k_2, \dots, k_n)$ tuples and finding the corresponding optimal c_i values for each $(k_1, k_2, \dots, k_n)$ tuple.

Next, for a given set of $(k_1, k_2, \dots, k_n)$ values, we find the corresponding optimal c_i values. For that, we introduce the Lagrangian function [149] for (7.7) as

$$\begin{aligned}\mathcal{L} = &-\sum_{i=1}^{n} k_i \frac{u_i}{u_i+\lambda_i}\frac{c_i}{c_i+\lambda_i} + (1-k_i)\frac{c_i}{c_i+\lambda_i+\frac{c_i\lambda_i}{s_i}} + \beta\left(\sum_{i=1}^{n} c_i - C\right) \\ &+\sum_{i=1}^{n}\eta_i((1-k_i)c_i-u_i) - \sum_{i=1}^{n}\nu_i c_i, \end{aligned} \tag{7.8}$$

where $\beta \geq 0$, $\eta_i \geq 0$ and $\nu_i \geq 0$. The KKT conditions are

$$\frac{\partial \mathcal{L}}{\partial c_i} = -\frac{u_i}{u_i+\lambda_i}\frac{\lambda_i}{(c_i+\lambda_i)^2} + \beta - \nu_i = 0, \tag{7.9}$$

for all i with $k_i = 1$, and

$$\frac{\partial \mathcal{L}}{\partial c_i} = -\frac{\lambda_i}{\left(c_i+\lambda_i+\frac{c_i\lambda_i}{s_i}\right)^2} + \beta + \eta_i - \nu_i = 0, \tag{7.10}$$

for all i with $k_i = 0$. Complementary slackness conditions are

$$\beta \left(\sum_{i=1}^{n} c_i - C \right) = 0, \tag{7.11}$$

$$\eta_i((1 - k_i)c_i - u_i) = 0, \tag{7.12}$$

$$\nu_i c_i = 0. \tag{7.13}$$

For given k_is with $k_i = 1$, we rewrite (7.9) as

$$(c_i + \lambda_i)^2 = \frac{1}{\beta - \nu_i} \frac{u_i \lambda_i}{u_i + \lambda_i}. \tag{7.14}$$

If $c_i > 0$, we have $\nu_i = 0$ from (7.13). Thus, we have

$$c_i = \left(\frac{1}{\sqrt{\beta}} \sqrt{\frac{u_i \lambda_i}{u_i + \lambda_i}} - \lambda_i \right)^+, \tag{7.15}$$

for all i with $k_i = 1$, where $(x)^+ = \max(x, 0)$. Similarly, for given k_is with $k_i = 0$, we rewrite (7.10) as

$$\left(c_i + \lambda_i + \frac{c_i \lambda_i}{s_i} \right)^2 = \frac{\lambda_i}{\beta + \eta_i - \nu_i}. \tag{7.16}$$

If $c_i > 0$, we have $\nu_i = 0$ from (7.13). Furthermore, if $c_i < u_i$, then we have $\eta_i = 0$ from (7.12). Otherwise, we have $c_i = u_i$ and $\eta_i \geq 0$ from (7.12). Thus, we have

$$c_i = \min \left(\frac{s_i}{s_i + \lambda_i} \left(\sqrt{\frac{\lambda_i}{\beta}} - \lambda_i \right)^+, u_i \right), \tag{7.17}$$

for all i with $k_i = 0$.

Note that $c_i > 0$ in (7.15) requires $\frac{1}{\lambda_i}\frac{u_i}{u_i+\lambda_i} > \beta$ which also implies that if $\frac{1}{\lambda_i}\frac{u_i}{u_i+\lambda_i} \leq \beta$, then we must have $c_i = 0$. Similarly, $c_i > 0$ in (7.17) requires $\frac{1}{\lambda_i} > \beta$ which also implies that if $\frac{1}{\lambda_i} \leq \beta$, then we must have $c_i = 0$. Thus, for given k_is, we observe that the optimal rate allocation policy for the cache is a *threshold policy* in which the optimal update rates are equal to zero when the file update rates λ_is are too large, i.e., when the files are updated too frequently at the source. In the optimal policy, the total update rate constraint for the cache, i.e., $\sum_{i=1}^{n} c_i \leq C$, should be satisfied with equality as the objective function in (7.7) is an increasing function of c_i.

For given k_is and u_is, we define ϕ_i as

$$\phi_i = \begin{cases} \frac{1}{\lambda_i}\frac{u_i}{u_i+\lambda_i}, & \text{if } k_i = 1, \\ \frac{1}{\lambda_i}, & \text{if } k_i = 0. \end{cases} \tag{7.18}$$

Similar to [101, Lemma 3], for given k_is and u_is, if $c_i > 0$ for some i, then we have $c_j > 0$ for all j with $\phi_j \geq \phi_i$.

Next, for a given set of k_is and u_is, we find the optimal c_is. First, we obtain ϕ_i from (7.18). We initially assume that $c_i < u_i$ for all i with $k_i = 0$, i.e., c_i in (7.17) is equal to $\frac{s_i}{s_i+\lambda_i}\left(\sqrt{\frac{\lambda_i}{\beta}} - \lambda_i\right)^+$. Then, we rewrite (7.15) and (7.17) as

$$c_i = \begin{cases} \frac{\lambda_i}{\sqrt{\beta}}\left(\sqrt{\phi_i} - \sqrt{\beta}\right)^+, & \text{if } k_i = 1, \\ \frac{s_i}{s_i+\lambda_i}\frac{\lambda_i}{\sqrt{\beta}}\left(\sqrt{\phi_i} - \sqrt{\beta}\right)^+, & \text{if } k_i = 0. \end{cases} \tag{7.19}$$

As we discussed earlier, in the optimal policy, we must have $\sum_{i=1}^{n} c_i = C$. Similar to the solution method in [101], we solve $\sum_{i=1}^{n} c_i = C$ for β by assuming that $\phi_i \geq \beta$ for all i, i.e., by ignoring $(\cdot)^+$ in (7.19). Then, we compare the smallest ϕ_i with β. If the smallest ϕ_i is larger than or equal to β, it implies that $c_i > 0$ for all i as we assumed before, and we have obtained c_i values for given k_is. If the smallest ϕ_i is smaller than β, it implies that the corresponding c_i was negative and it must be chosen as zero. In this case, we choose $c_i = 0$ for the smallest ϕ_i. Then, we repeat this process again until the smallest ϕ_i among the remaining c_is satisfies $\phi_i \geq \beta$.

Finally, when we find all c_i values, we go back to our initial assumption which is $c_i < u_i$ for all i with $k_i = 0$ and check whether it holds or not. We define the set $S = \{i | c_i > u_i, k_i = 0\}$. If we have $c_i < u_i$ for all i with $k_i = 0$, i.e., when S is empty, then we obtain the optimal c_i values. If we have $c_i > u_i$ for some i with $k_i = 0$, then in the optimal policy, we have $c_i = u_i$ for all $i \in S$. Then, for remaining c_is with $i \notin S$, we repeat this process again with the remaining total update rate, i.e., $C - \sum_{i \in S} u_i$, until we have $c_i \leq u_i$ for all i with $k_i = 0$.

7.5 Numerical Results

In this section, we provide two numerical results for the optimal solution obtained in Section 7.4 for $n = 8$. For these results, we consider the update arrival rates at the source $\lambda_i = bq^i$ with $q = 0.7$ such that $\sum_{i=1}^{n} \lambda_i = 10$. We take the file request rates at the user $u_i = dr^i$ with $r = 0.8$ such that $\sum_{i=1}^{n} u_i = 20$. Finally, we take the

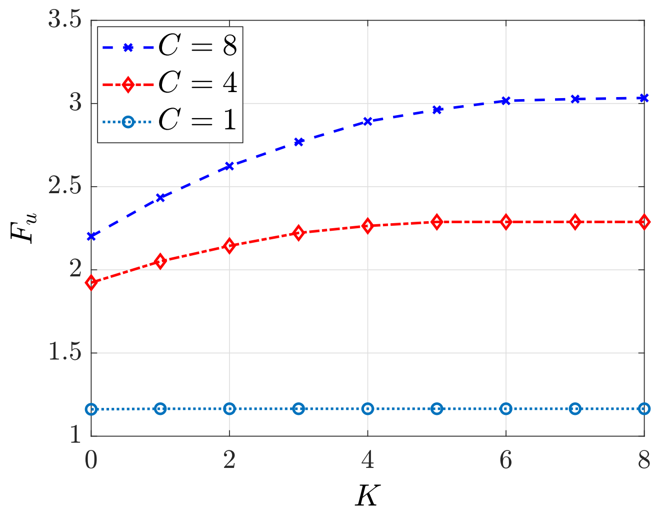


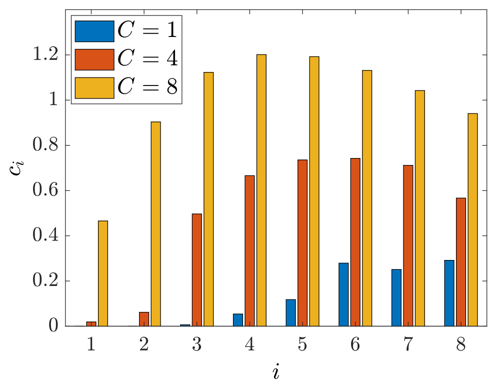
Figure 7.3: Total freshness of the user F_u with respect to the cache capacity K when the total cache update rate is $C = 1, 4, 8$.

file transmission rates at the source $s_i = hp^i$ with $p = 1.25$ such that $\sum_{i=1}^{n} s_i = 3$.

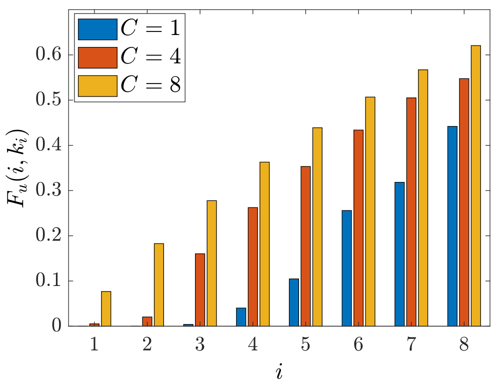
In the first example, we increase the cache capacity K from 0 to n when the total update rate at the cache is $C = 1, 4, 8$. We observe in Fig. 7.3 that when the total cache update rate is small, i.e., when $C = 1$, increasing the cache capacity K does not improve the freshness of the user much, i.e., F_u stays constant for $K \geq 1$. As the total cache update rate C is too low, if a file is stored at the cache, the user gets obsolete versions of the file most of the time. Thus, we observe that even though the cache capacity is high, the optimal policy is to cache only one file. In this case, the cache mostly forwards the update requests from the user to the source, i.e., the cache behaves like a relay node. When the total cache update rate increases, i.e., when $C = 4$, we observe in Fig. 7.3 that increasing the cache capacity K increases the user freshness up to $K = 5$ and does not improve it for $K > 5$. Similarly, when $C = 8$, we observe in Fig. 7.3 that the user freshness increases with the cache capacity. In this case, as the total cache update rate is high enough, the optimal policy is to cache every file.

In the second example, we consider the same system as in the first example, but we take $K = n$ and find the optimal caching status for each file k_i and the corresponding file update rates at the cache c_i. When $C = 1$, the optimal policy is to cache only the 6th file, i.e., $k_6 = 1$ and $k_i = 0$ for $i \neq 6$. When $C = 4$, the optimal caching status is $k_i = 1$, for $i = 3, 4, 5, 6, 7$ and $k_i = 0$, otherwise. Thus, we observe that the files that change too fast at the source are not cached. Furthermore, as the file transmission rate of the 8th file is too high, we see that the 8th file is not cached. When $C = 8$, it is optimal to cache every file, i.e., $k_i = 1$ for all i. Thus, when the total cache update rate is high enough, the optimal policy is to cache every file as caching helps user to avoid the transmission time between the source and the user. However, when the total cache update rate is limited, the optimal policy is not to cache the files that are frequently updated at the source or the files that have smaller transmission times.

The optimal file update rate of the cache c_i is shown in Fig. 7.4(a). When $C = 1$, i.e., when the total cache update rate is too small, the first two files which are updated at the source most frequently are not updated by the cache, i.e., $c_1 = c_2 = 0$. Furthermore, we observe in Fig. 7.4(a) that the file update rates at the cache initially increase with the file indices up to $i = 6$ when $C = 4$ and up to $i = 4$ when $C = 8$, and then decrease for the remaining files. The freshness of the files at the user $F_u(i, k_i)$ is shown in Fig. 7.4(b). We see in Fig. 7.4(b) that the files that change slowly at the source have higher file freshness at the user even though the file update rates at the cache get lower. We observe that increasing the total cache update rate C improves



(a)



(b)

Figure 7.4: (a) Update rate allocation at the cache for each file, and (b) the corresponding freshness $F_u(i, k_i)$, when $C = 1, 4, 8$.

the freshness of the files. In addition, we note that the freshness improvement is higher on the rapidly changing files compared to the others.

7.6 Conclusion

In this chapter, we considered a caching system with limited storage capacity for the cache. Here, the user can obtain the files from the cache, but the received files can be outdated if the cache does not have the freshest version of the files at the time of an update request. The user can also obtain the freshest files directly from the source at the expense of an additional transmission time between the user and the source. In this work, we determined the optimal file caching status of files, and their corresponding age-optimal file update rates at the cache. We observed that when the total update rate of the cache is limited, it is optimal to obtain some of the files that change frequently at the source and have small transmission times directly from the source, and store the remaining files at the cache.

CHAPTER 8

Freshness in Gossiping Networks

8.1 Introduction

In this chapter, our aim is to study information timeliness in arbitrarily connected and structured gossip networks using the binary freshness metric. Works that are closely related to our work are [99,101,102,131,132]. The binary freshness metric has been studied for serially connected caching systems in [99,101] (also Chapters 6 and 7 in this thesis) and for parallel connected caching systems in [102]. References [99,101, 102] maximize information freshness at the end-nodes in caching systems by using alternating maximization methods. Different from [99, 101, 102], in this chapter, we employ the stochastic hybrid systems (SHS) approach and develop a general method that enable us to characterize the binary freshness in *arbitrarily* connected networks. We explore gossiping strategies among the end-nodes and study freshness in structured gossip networks, i.e., disconnected, ring and fully connected network topologies. Reference [131] uses the SHS approach to characterize the version age and finds the scaling of version age in structured gossip networks. Reference [132] improves the scaling of version age by introducing *clustering* to gossip networks.

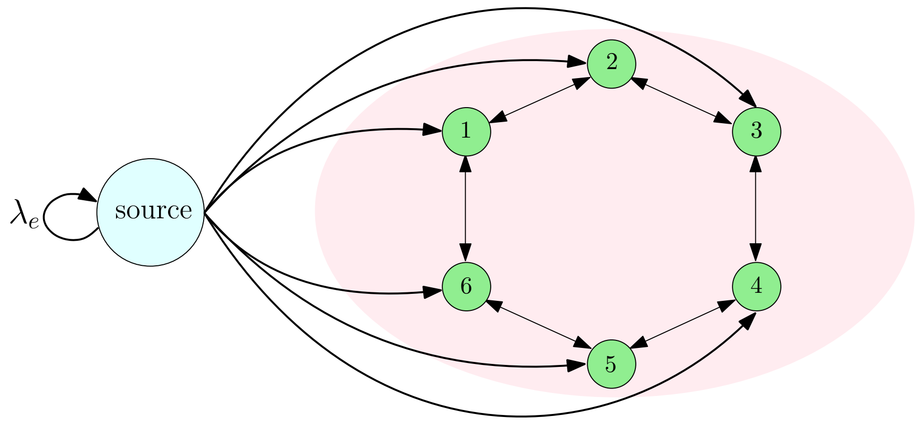


Figure 8.1: Gossip network model consisting of a source represented by the blue node, and the users represented by the green nodes. Here, users form a ring network. Other network topologies are shown in Fig. 8.2.

Here, we develop the binary freshness counterpart of the works [131, 132]. As the binary freshness metric and encompassing mathematics are different than those in version age, our work is distinct from works [131, 132].

In this work, by using the SHS approach, we first provide a way to characterize binary freshness in arbitrarily connected networks. Then, we consider binary freshness in structured gossip networks, such as disconnected, ring and fully connected networks; see Figs. 8.1-8.2. We show that when the number of nodes becomes large, the binary freshness of a node decreases down to 0 with rate n^{-1} for the disconnected and ring networks, but the freshness is strictly larger for the ring networks. For the fully connected networks, when the update rates of the source and the end-nodes are sufficiently large, the binary freshness of a node decreases down to 0 slower than the other network types, indicating that increased connectivity among the nodes improves the freshness. Finally, we find the binary freshness in clustered gossip networks, where each cluster consists of a structured gossip network and is connected to the source through a designated access node, i.e., a cluster head. We numerically

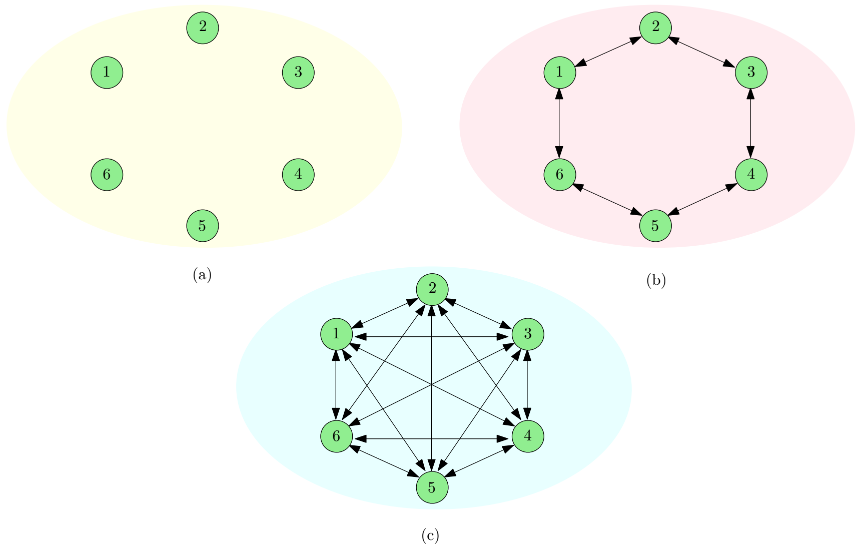


Figure 8.2: (a) Disconnected, (b) ring, and (c) fully connected network topologies with $n = 6$ end-nodes.

find the optimal number of nodes in each cluster for given update rates of the source, cluster heads, and the end-nodes.

8.2 Freshness in Arbitrarily Connected Networks

In this section, we consider freshness in gossip networks with arbitrary topologies. System model considered in this section is complementary to the models in [131, 132], in that we use the binary freshness metric to characterize timeliness in gossip networks as opposed to the version age metric used in [131,132]. In our model, there is a single source node that is updated as a Poisson process with rate λ_e. This source node updates the nodes in the system with a total rate of λ. There are n nodes in the system, which are denoted by the set $\mathcal{N} \triangleq \{1, 2, \dots, n\}$. Nodes in the system implement gossiping as in [131,132] to distribute update packets among each other. The total update rate of a node is λ, and each node in $\mathcal{N}$ uniformly distributes its update rate among its neighbors.

We first characterize the binary freshness in arbitrarily connected gossip networks. Following the analysis for the version age in [131,132], we write the freshness F_i in terms of F_S, which denotes the freshness of a subset S and it is given by $F_S(t) \triangleq \max_{j\in S} F_j(t)$. We note that here we have a maximum operator as opposed to a minimum operator since our freshness metric takes the maximum value when the information at the receiver node is the freshest.

As in [131], we denote the total update rate from node i into set S by $\lambda_i(S)$, i.e., $\lambda_i(S) = \sum_{j\in S} \lambda_{ij}$ when $i \notin S$. Similarly, $\lambda_s(S)$ denotes the total update rate from the source into set S. Set $N(S)$ denotes the set of updating neighbors of S given by $N(S) = \{i \in \{1,\dots,n\} : \lambda_i(S) > 0\}$.

In Theorem 8.1, we characterize the freshness in arbitrarily connected gossip networks. This result is the binary freshness equivalent of the version age result in [131, Thm. 1].

Theorem 8.1 *The average freshness of a set S, $F_S = \lim_{t\to\infty} \mathbb{E}[F_S(t)]$, for $S \subseteq \mathcal{N}$ is given by*

$$F_S = \frac{\lambda_s(S) + \sum_{i\in N(S)} \lambda_i(S) F_{S\cup\{i\}}}{\lambda_e + \lambda_s(S) + \sum_{i\in N(S)} \lambda_i(S)}. \tag{8.1}$$

Proof: A state transition in the system happens when a node i updates node j. We first present the possible state transitions. Let $\mathcal{L}$ denote the set of transitions as in [131]. Accordingly,

$$\mathcal{L} = \{(s,s)\} \cup \{(s,j) : j \in \mathcal{N}\} \cup \{(i,j) : i,j \in \mathcal{N}\}, \tag{8.2}$$

where the first transition occurs when the source generates a new update, the second set of transitions occur when the source node directly updates an end-node in $\mathcal{N}$, and the third set of transitions occur when an end-node updates another end-node. In our case, different than [131], the freshness vector evolves as

$$F'_k = \begin{cases} 0, & i = j = s, k \in \mathcal{N}, \\ 1, & i = s, j = k \in \mathcal{N}, \\ \max(F_i, F_j), & i \in \mathcal{N}, j = k \in \mathcal{N}, \\ F_k, & \text{otherwise}, \end{cases} \tag{8.3}$$

where F'_k is the freshness of node k after a transition. In (8.3), freshness takes the minimum value of 0 when node k has stale information while it takes the maximum value of 1 when node k has fresh information, i.e., the same information as the source. After the (s,s) transition, the freshness of the set S becomes 0 as all the nodes in set S become out-of-date. For other transitions (i,j), given that $j \in S$, we have

$$F'_S = \max_{k \in S} F'_k = \max_{k \in S \cup \{i\}} F_k = F_{S \cup \{i\}}. \tag{8.4}$$

We note that when $i = s$, (8.4) implies that $F'_S = 1$. If $j \notin S$, the freshness of set S is unchanged after transition (i,j), i.e., $F'_S = F_S$. Using (8.3) and (8.4) and following similar steps as in [131] yields the result. ■

8.3 Sample Freshness Evaluations

To demonstrate the application of (8.1), in what follows, we consider simple network examples and characterize the average binary freshness experienced by the end-nodes.

First, we consider the network in Fig. 8.3(a). The information at the source is updated with rate λ_e. The source sends updates to the cache with rate λ_d and the cache sends updates to node 1 with rate λ_c. By noting that only the cache updates node 1 and there is no direct link from the source to node 1, we have $N(1) = C$ and $\lambda_s(1) = 0$. Thus, from (8.1), we find $F_1 = \frac{\lambda_c F_{\{1\cup C\}}}{\lambda_e+\lambda_c}$. As the subset $\{1 \cup C\}$ gets updates only from the source, we have $N(\{1\cup C\}) = \emptyset$ and $\lambda_s(\{1\cup C\}) = \lambda_d$. Thus, $F_{\{1\cup C\}}$ becomes $\frac{\lambda_d}{\lambda_d+\lambda_e}$. By inserting $F_{\{1\cup C\}}$ into F_1,

$$F_1 = \frac{\lambda_c}{\lambda_c + \lambda_e}\frac{\lambda_d}{\lambda_d + \lambda_e}. \tag{8.5}$$

Note that, in this example, the nodes are connected serially and the freshness expression in (8.5) is the same as [101, (11)].

Second, we consider the network in Fig. 8.3(b) where we have a source, two caches connected in parallel, and a user. We want to find the freshness at node 1,

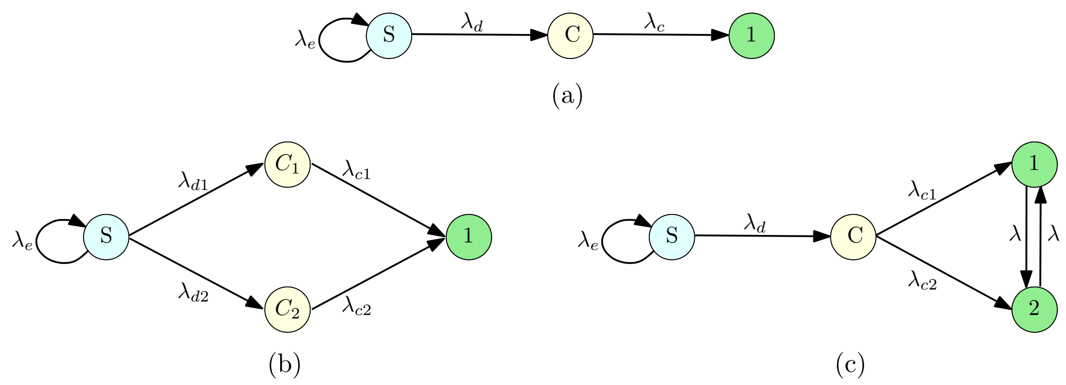


Figure 8.3: Freshness of information in (a) a serially connected network, (b) a parallel connected network, and (c) an arbitrarily connected network.

i.e., F_1. Since $N(1) = \{C_1, C_2\}$ and $\lambda_s(1) = 0$, F_1 is equal to

$$F_1 = \frac{\lambda_{c1} F_{\{1\cup C_1\}} + \lambda_{c2} F_{\{1\cup C_2\}}}{\lambda_e + \lambda_{c1} + \lambda_{c2}}. \tag{8.6}$$

Then, we find $F_{\{1\cup C_1\}} = (\lambda_{d1} + \lambda_{c2} F_{\{1\cup C_1\cup C_2\}})/(\lambda_e + \lambda_{d1} + \lambda_{c2})$, $F_{\{1\cup C_2\}} = (\lambda_{d2} + \lambda_{c1} F_{\{1\cup C_1\cup C_2\}})/(\lambda_e + \lambda_{d2} + \lambda_{c1})$, $F_{\{1\cup C_1\cup C_2\}} = (\lambda_{d1} + \lambda_{d2})/(\lambda_e + \lambda_{d1} + \lambda_{d2})$. Inserting $F_{\{1\cup C_1\}}$, $F_{\{1\cup C_2\}}$, and $F_{\{1\cup C_1\cup C_2\}}$ back into (8.6), we obtain

$$\begin{aligned} F_1 =& \frac{(\lambda_{d1} + \lambda_{d2})(\lambda_{c1} + \lambda_{c2})}{(\lambda_e + \lambda_{d1} + \lambda_{d2})(\lambda_e + \lambda_{c1} + \lambda_{c2})} - \frac{\lambda_e}{(\lambda_e + \lambda_{d1} + \lambda_{d2})(\lambda_e + \lambda_{c1} + \lambda_{c2})} \times \\ & \left(\frac{\lambda_{c2}\lambda_{d1}}{\lambda_e + \lambda_{c1} + \lambda_{d2}} + \frac{\lambda_{c1}\lambda_{d2}}{\lambda_e + \lambda_{c2} + \lambda_{d1}} \right). \end{aligned} \tag{8.7}$$

Note that, in this example, the caches are connected in parallel and the freshness expression in (8.7) is the same as [102, (21)] but its derivation is much simpler here.

Third, we consider the network in Fig. 8.3(c). In this example, we want to find the freshness at node 1, i.e., F_1. Since $N(1)=\{2,C\}$ and $\lambda_s(1)=0$, F_1 is equal to

$$F_1=\frac{\lambda_{c1}F_{\{1\cup C\}}+\lambda F_{\{1\cup 2\}}}{\lambda_e+\lambda_{c1}+\lambda}. \tag{8.8}$$

Similarly, we find $F_{\{1\cup C\}}$, $F_{\{1\cup 2\}}$, and $F_{\{1\cup 2\cup C\}}$. By using these in (8.8), we obtain

$$F_1=\frac{\lambda_{c1}}{\lambda_e+\lambda_{c1}+\lambda}\frac{\lambda_d+\lambda\frac{\lambda_d}{\lambda_d+\lambda_e}}{\lambda_e+\lambda_d+\lambda}+\frac{\lambda}{\lambda_e+\lambda_{c1}+\lambda}\frac{(\lambda_{c1}+\lambda_{c2})\frac{\lambda_d}{\lambda_d+\lambda_e}}{\lambda_e+\lambda_{c1}+\lambda_{c2}}. \tag{8.9}$$

Note that, in this example, the nodes are arbitrarily connected, i.e., the network includes both serial and parallel connections. Prior to this work, the freshness expression was known only for serially connected networks [101] and parallel relay networks [102]. Thus, with the method developed here, we are able to find the freshness expression for arbitrarily connected networks.

8.4 Freshness in Structured Gossip Networks

In this section, we consider information freshness in structured networks. We consider three different network structures regarding the connectivity among the end-nodes: disconnected, ring and fully connected networks. In all cases, there is a single source node and n end-nodes. The information at the source node is updated with rate λ_e. The source node updates each end-node with rate $\frac{\lambda}{n}$. Each end-node updates its immediate neighbors with a total λ update rate equally distributed over the neighbors. In what follows, we denote $\rho \triangleq \frac{\lambda_e}{\lambda}$.

8.4.1 Disconnected Networks

In this network, the end-nodes are not connected to each other and only the source node updates the end-nodes with rate $\frac{\lambda}{n}$ for each node; this network is obtained when Fig. 8.2(a) is inserted into Fig. 8.1. Thus, for an arbitrary node S_1, we have $\lambda_s(S_1) = \frac{\lambda}{n}$ and $N(S_1) = \emptyset$. Then, from (8.1), we have

$$F_{S_1} = \frac{1}{1+n\rho}. \tag{8.10}$$

Hence, the freshness of a node goes to 0 as $\frac{1}{n}$ with the network size n. This is stated formally in Theorem 8.2 below.

Theorem 8.2 *In a disconnected network, the average freshness of a single node decreases down to* 0 *as* $\frac{1}{\rho}n^{-1}$.

8.4.2 Ring Networks

In this network, the end-nodes are connected to each other as a bidirectional ring where each node updates its two neighbors with rate $\frac{\lambda}{2}$ each, and the source node updates the end-nodes with rate $\frac{\lambda}{n}$ for each node; this is the network in Fig. 8.1. Here, subset S_j denotes any arbitrary j adjacent nodes. In particular, if $S_j = \{1,\dots,j\}$, then we have $\lambda_s(S_j) = \frac{j\lambda}{n}$, and $N(S_j) = \{j+1, n\}$ (see Fig. 8.2(b)). Thus, from (8.1), we have

$$F_{S_j} = \frac{\frac{j\lambda}{n} + \lambda F_{S_{j+1}}}{\lambda_e + \lambda + \frac{j\lambda}{n}}, \tag{8.11}$$

for $j = 1, \ldots, n-1$, and $F_{S_n} = \frac{1}{1+\rho}$. Theorem 8.3 below states the form of the freshness of a node for large n. We note that the freshness in a ring network in Theorem 8.3 is larger than the freshness in a disconnected network in Theorem 8.2.

Theorem 8.3 *In a ring network, the average freshness of a single node decreases down to* 0 *as* $\left(\frac{1}{\rho}+\frac{1}{\rho^2}\right)n^{-1}$.

Proof: From (8.11), we write the freshness of a node F_{S_1} as

$$F_{S_1} = \sum_{i=1}^{n-1} a_i^{(n)} + \frac{n}{n-1} a_{n-1}^{(n)} F_{S_n}, \tag{8.12}$$

where $a_i^{(n)}$ is given for $i = 1, \ldots, n-1$ as

$$a_i^{(n)} = \frac{i}{n} \prod_{j=1}^{i} \frac{1}{1+\rho+\frac{j}{n}} = \frac{i}{n}\frac{1}{(1+\rho)^i} \prod_{j=1}^{i} \frac{1}{1+\frac{1}{1+\rho}\frac{j}{n}}. \tag{8.13}$$

From [132, eqns. (5)-(6)], the product term in (8.13) can be approximated as $e^{-\frac{i^2}{2(1+\rho)n}}$, and $a_i^{(n)}$ in (8.13) can be approximated as $a_i^{(n)} \approx \frac{i}{n}\frac{1}{(1+\rho)^i} e^{-\frac{i^2}{2(1+\rho)n}} \approx \frac{i}{n}\frac{1}{(1+\rho)^i}$. Then, since $\sum_{i=1}^{\infty} i \frac{1}{(1+\rho)^i} = \frac{1}{\rho} + \frac{1}{\rho^2}$, and the second term in (8.12) is negligible, we have $F_{S_1} \approx \sum_{i=1}^{n-1} a_i^{(n)} \approx \left(\frac{1}{\rho}+\frac{1}{\rho^2}\right)\frac{1}{n}$. ■

8.4.3 Fully Connected Networks

In this network, the end-nodes are fully connected, and each end-node is updated by each of the remaining end-nodes with rate $\frac{\lambda}{n-1}$ and by the source with rate $\frac{\lambda}{n}$. Let S_j denote a subset of any arbitrary j nodes. Since $\lambda_s(S_j) = \frac{j\lambda}{n}$ and $N(S_j) = \mathcal{N} \setminus S_j$.

Thus, from (8.1), we have

$$F_{S_j} = \frac{\frac{j\lambda}{n} + \frac{j(n-j)\lambda}{n-1} F_{S_{j+1}}}{\lambda_e + \frac{j\lambda}{n} + \frac{j(n-j)\lambda}{n-1}}, \tag{8.14}$$

for $j = 1, \ldots, n-1$, and $F_{S_n} = \frac{1}{1+\rho}$. Theorem 8.4 below gives the freshness of a node for large n. We note that the freshness in a fully connected network in Theorem 8.4 is larger than the freshness in disconnected or ring networks in Theorems 8.2 and 8.3. Thus, connectedness improves freshness in gossip networks.

Theorem 8.4 *In a fully connected network, the average freshness of a single node decreases down to* 0 *as* $n^{-\rho}$ *when* $0 < \rho < 1$*; as* $\frac{\log(n)}{n}$ *when* $\rho = 1$*; and as* n^{-1} *when* $\rho > 1$.

Proof: Using (8.14), we write the freshness of a node F_{S_1} as

$$F_{S_1} = \sum_{i=1}^{n-1} b_i^{(n)} + \frac{n}{n-1} b_{n-1}^{(n)} F_{S_n}, \tag{8.15}$$

where $b_i^{(n)}$ is given for $i = 1, \ldots, n-1$ as

$$b_i^{(n)} = \frac{1}{1 + \frac{n\rho}{i} + \frac{n}{n-1}(n-i)} \prod_{j=1}^{i-1} \frac{1}{1 + \frac{(n-1)\rho}{(n-j)j} + \frac{n-1}{n}\frac{1}{n-j}}. \tag{8.16}$$

Then, we have

$$-\log(b_i^{(n)}) = \log\left(1 + \frac{n\rho}{i} + \frac{n}{n-1}(n-i)\right) + \sum_{j=1}^{i-1} \log\left(1 + \frac{(n-1)\rho}{(n-j)j} + \frac{n-1}{n}\frac{1}{n-j}\right). \tag{8.17}$$

Inside the second $\log(\cdot)$ in (8.17), $\frac{(n-1)\rho}{(n-j)j} = \frac{(n-1)\rho}{n}\left(\frac{1}{j} + \frac{1}{n-j}\right)$. For large n, we have $\frac{n-1}{n} \approx 1$, and for small x, we have $\log(1+x) \approx x$, then the second term in (8.17) becomes $\sum_{j=1}^{i-1} \log\left(1+\frac{(n-1)\rho}{(n-j)j} + \frac{n-1}{n}\frac{1}{n-j}\right) \approx (1+\rho)\sum_{j=1}^{i-1}\frac{1}{n-j} + \rho\sum_{j=1}^{i-1}\frac{1}{j}$. For large i, $\sum_{j=1}^{i-1}\frac{1}{j} \approx \log i$ and $\sum_{j=1}^{i-1}\frac{1}{n-j} \approx \log n - \log(n-i)$. Thus, $-\log(b_i^{(n)})$ in (8.17) becomes

$$-\log(b_i^{(n)}) \approx c + (1+\rho)\log n - \rho\log(n-i) + \rho\log i, \tag{8.18}$$

where $c = \log\left(1 + \frac{\rho}{i} + \frac{1+\rho}{n-i}\right)$, and $\log(1) \leq c \leq \log(2+\rho)$. Thus, we rewrite $b_i^{(n)}$ in (8.16) as $b_i^{(n)} \approx \frac{d}{n}\left(\frac{1}{i} - \frac{1}{n}\right)^{\rho}$ where $d = e^{-c}$, which is a constant between 1 and $1/(2+\rho)$. Since the last term in (8.15) is negligible for large n, we have

$$F_{S_1} = \sum_{i=1}^{n-1} b_i^{(n)} \approx \frac{d}{n}\sum_{i=1}^{n-1}\left(\frac{1}{i} - \frac{1}{n}\right)^{\rho}. \tag{8.19}$$

By noting that $\frac{1}{2^\rho}\frac{1}{i^\rho} \leq \left(\frac{1}{i} - \frac{1}{n}\right)^{\rho}$ for $1 \leq i \leq \frac{n}{2}$, we have

$$\frac{1}{2^\rho}\sum_{i=1}^{n/2}\frac{1}{i^\rho} \leq \sum_{i=1}^{n-1}\left(\frac{1}{i} - \frac{1}{n}\right)^{\rho} \leq \sum_{i=1}^{n-1}\frac{1}{i^\rho}. \tag{8.20}$$

By using Riemann sum, for large n, we approximate $\sum_{i=1}^{n}\frac{1}{i^\rho}$ as

$$\sum_{i=1}^{n}\frac{1}{i^\rho} \approx \begin{cases} \frac{n^{1-\rho}}{1-\rho}, & \text{when } 0 < \rho < 1, \\ \log n, & \text{when } \rho = 1, \\ \text{constant}, & \text{when } \rho > 1. \end{cases} \tag{8.21}$$

Thus, F_{S_1} in (8.19) becomes

$$F_{S_1} \approx \frac{d}{n}\sum_{i=1}^{n-1}\left(\frac{1}{i}-\frac{1}{n}\right)^{\rho} \approx \begin{cases} \frac{1}{n^\rho}, & \text{when } 0<\rho<1, \\ \frac{\log n}{n}, & \text{when } \rho=1, \\ \frac{1}{n}, & \text{when } \rho>1, \end{cases} \tag{8.22}$$

which completes the proof. ■

8.5 Freshness in Clustered Gossip Networks

In this section, similar to [132], we consider clustered networks, in which each cluster has an associated cluster head that receives updates from the source and forwards them to the corresponding nodes in its own cluster. The total of n nodes are divided into m clusters, with k nodes in each cluster, thus $n = km$. Here, the source updates the cluster heads with a total update rate of λ_s (thus, $\frac{\lambda_s}{m}$ update rate per cluster head); each cluster head updates its end-nodes with a total update rate of λ_c (thus, $\frac{\lambda_c}{k}$ update rate per end-node); and each end-node updates its immediate neighbors with a total update rate of λ. Since the nodes (among themselves) and the cluster heads (among themselves) are symmetrical, freshness experienced by each end-node is statistically identical. Thus, in the following analysis, we consider the freshness of a typical node in an arbitrary cluster. In addition to the aforementioned definitions of $\lambda_i(S)$ for $i = \{1, \dots, n\}$ and $N(S)$ for an arbitrary subset S of the nodes, in what follows, $\lambda_c(S)$ denotes the total update rate of a cluster head into the subset S.

In Theorem 8.5, we characterize the freshness in clustered gossip networks.

Theorem 8.5 *In a clustered network with m clusters of k nodes each, the freshness of a subset S is given by*

$$F_S = \frac{\lambda_c(S)F_c + \sum_{i\in N(S)} \lambda_i(S) F_{S\cup\{i\}}}{\lambda_e + \lambda_c(S) + \sum_{i\in N(S)} \lambda_i(S)}, \tag{8.23}$$

with $F_c = \frac{\lambda_s}{\lambda_s + m\lambda_e}$.

The proof of Theorem 8.5 follows from the application of Theorem 8.1 to clustered networks. In the following subsections, we characterize the information freshness recursions for the disconnected, ring and fully connected cluster models.

8.5.1 Disconnected Clusters

The overall disconnected clustered network behaves like a two hop multicast network. In the first hop, the source sends the updates to m cluster heads. In the second hop, each cluster head sends updates to k nodes within its cluster. Different from the multicast network studies in [19–22, 24], here, we consider freshness of information at the end-nodes by random gossiping without applying any central control over the update flows.

Let S_j denote an arbitrary subset of j nodes in a cluster. Since nodes are disconnected, we have $N(S_1) = \emptyset$ and thus, we write the freshness of a single node as

$$F_{S_1} = \frac{\lambda_c F_c}{\lambda_c + k\lambda_e} = \frac{\lambda_c}{\lambda_c + k\lambda_e}\frac{\lambda_s}{\lambda_s + m\lambda_e}. \tag{8.24}$$

In this network, a single node is serially connected to the source via its cluster head. Thus, the freshness in (8.24) has the same form as in (8.5). Next, we consider how fast the freshness goes down to 0 as the number of nodes, n, grows large.

Theorem 8.6 *In a clustered disconnected network, the freshness of a node decreases down to* 0 *as* $\frac{\lambda_c \lambda_s}{\lambda_e^2} n^{-1}$.

We note that even with the use of clusters, the freshness of a node still decreases down to 0 with rate n^{-1}. When $\lambda_c > \lambda_e$, the convergence rate is slightly improved, i.e., it increases to $\frac{\lambda_c \lambda_s}{\lambda_e^2} n^{-1}$ in Theorem 8.6 from $\frac{\lambda_s}{\lambda_e}$ in Theorem 8.2. When $\lambda_c < \lambda_e$, the freshness of a node converges to 0 faster. That is, the presence of cluster heads may hurt information freshness in a disconnected network when the update rates of cluster heads are not sufficiently large.

8.5.2 Ring Clusters

With ring clusters, there are two nodes that update the subset S_j with a total rate λ. From (8.23), we write

$$F_{S_j} = \frac{\frac{j\lambda_c}{k} F_c + \lambda F_{S_{j+1}}}{\lambda_e + \frac{j\lambda_c}{k} + \lambda}, \tag{8.25}$$

for $j = 1, \ldots, k-1$, and we have $F_{S_k} = \frac{\lambda_c}{\lambda_c + \lambda_e} \frac{\lambda_s}{\lambda_s + m\lambda_e}$. For $j = k$, the network simply becomes a two hop network where the first hop is from the source to the cluster head and the second hop is from the cluster head to the entire cluster.

8.5.3 Fully Connected Clusters

With fully connected clusters, each node is connected to other $k-1$ nodes with rates $\frac{\lambda}{k-1}$. Let S_j denote an arbitrary j-node subset in a cluster. Subset S_j receives updates from the cluster head with rate $\frac{j\lambda_c}{k}$ and from each of the remaining $k-j$ nodes in the same cluster with rate $\frac{\lambda}{k-1}$. With these, from (8.23), we write

$$F_{S_j} = \frac{\frac{j\lambda_c}{k}F_c + \frac{j(k-j)\lambda}{k-1}F_{S_{j+1}}}{\lambda_e + \frac{j\lambda_c}{k} + \frac{j(k-j)\lambda}{k-1}}, \tag{8.26}$$

for $j = 1, \ldots, k-1$, and we have $F_{S_k} = \frac{\lambda_c}{\lambda_c+\lambda_e}\frac{\lambda_s}{\lambda_s+m\lambda_e}$.

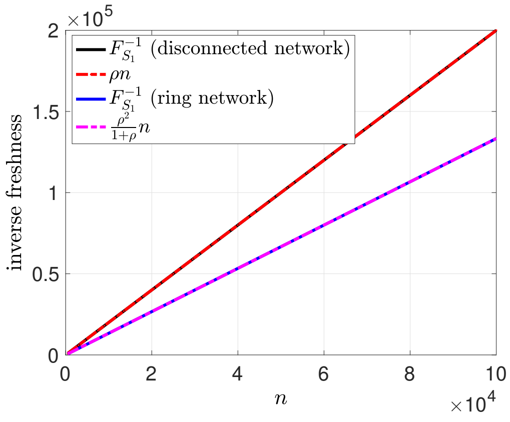
In the following section, we provide numerical results for the scaling of freshness in large gossip networks, as well as for the optimal selection of m and k for clustered network models with different λ, λ_s, and λ_c values.

8.6 Numerical Results

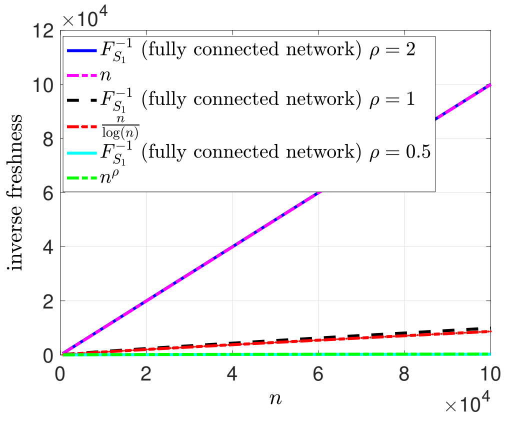
8.6.1 Numerical Results for Large Gossip Networks

In this subsection, we provide numerical results for the scaling of information freshness in large gossip networks. For all network types, since information freshness decreases to 0, we consider the scaling of the inverse freshness function of a node, i.e., $F_{S_1}^{-1} = \frac{1}{F_{S_1}}$, and vary n in between 500 to $100,000$.

In the first numerical result, we consider disconnected and ring networks for $\lambda_e = 2$, and $\lambda = 1$, i.e., $\rho = 2$. In Fig. 8.4(a), we see that the scaling of inverse



(a)



(b)

Figure 8.4: Scaling of inverse freshness of a node (a) in disconnected and ring networks when $\rho = 2$ ($\lambda_e = 2$ and $\lambda = 1$), (b) in a fully connected network when $\rho = 2$ ($\lambda_e = 2$ and $\lambda = 1$), $\rho = 1$ ($\lambda_e = 1$ and $\lambda = 1$), and $\rho = 0.5$ ($\lambda_e = 0.5$ and $\lambda = 1$).

freshness of a node is $F_{S_1}^{-1} \approx \rho n$ for the disconnected network, and $F_{S_1}^{-1} \approx \frac{\rho^2}{1+\rho} n$ for the ring network. Although the inverse freshness increases linearly with n for both network types, we see in Fig. 8.4(a) that the slope of inverse freshness in the ring network is smaller, indicating a slower decrease to 0 for freshness. This verifies that increased connectedness improves freshness in gossip networks.

In the second numerical result, we consider a fully connected network when $\rho = 2$ ($\lambda_e = 2$, and $\lambda = 1$), $\rho = 1$ ($\lambda_e = 1$, and $\lambda = 1$), and $\rho = 0.5$ ($\lambda_e = 0.5$, and $\lambda = 1$). In Fig. 8.4(b), we see that when $\rho = 2$, i.e., when the update rates of a node and the source (both are λ) are smaller compared to the information change rate at the source λ_e, then the inverse freshness still scales linearly with n. When λ gets large (specifically, when $\lambda = \lambda_e$), inverse freshness of a node starts to scale with $\frac{n}{\log(n)}$ as shown in Fig. 8.4(b). When we further increase λ, i.e., when $\lambda > \lambda_e$ (or $\rho < 1$), the inverse freshness scales with n^{ρ}, i.e., sublinearly. Thus, when the update rates of the source and the end-nodes are large enough in the fully connected network, freshness can be improved from $\frac{1}{n}$ to $\frac{1}{n^\rho}$.

8.6.2 Numerical Results for Clustered Gossip Networks

In this subsection, we provide numerical results regarding the optimal cluster size selection k in clustered networks for the cases of disconnected, ring and fully connected clusters, for different update rates at the source, cluster heads and the nodes, when information change rate at the source is $\lambda_e = 1$ and the number of nodes is $n = 120$.

First, we take $\lambda_s = 1$, $\lambda_c = 1$, $\lambda = 1$. In Fig. 8.5(a), we see that the optimal cluster size is $k^* = 40$ in fully connected clusters; $k^* = 20$ in ring clusters; $k^* = 10$ or $k^* = 12$ in disconnected clusters. We observe that the optimal cluster size increases as the connectivity among the nodes increases. Further, the achievable freshness increases with the connectivity within the clusters.

Second, we consider $\lambda_s = 10$, $\lambda_c = 1$, $\lambda = 1$, i.e., the update rate of the source is increased compared to the setting in Fig. 8.5(a). In Fig. 8.5(b), we see the trade-off between the number of clusters and the number of nodes in a cluster. Even though we increase the update rate of the source, since it is not large enough, freshness initially increases with the cluster size k as the total number of clusters decreases. That is, the source can send updates to each cluster more efficiently with increasing k. On the other hand, since the update rates at the cluster heads are also limited, further increasing k starts to decrease the freshness at the end-nodes. Thus, there

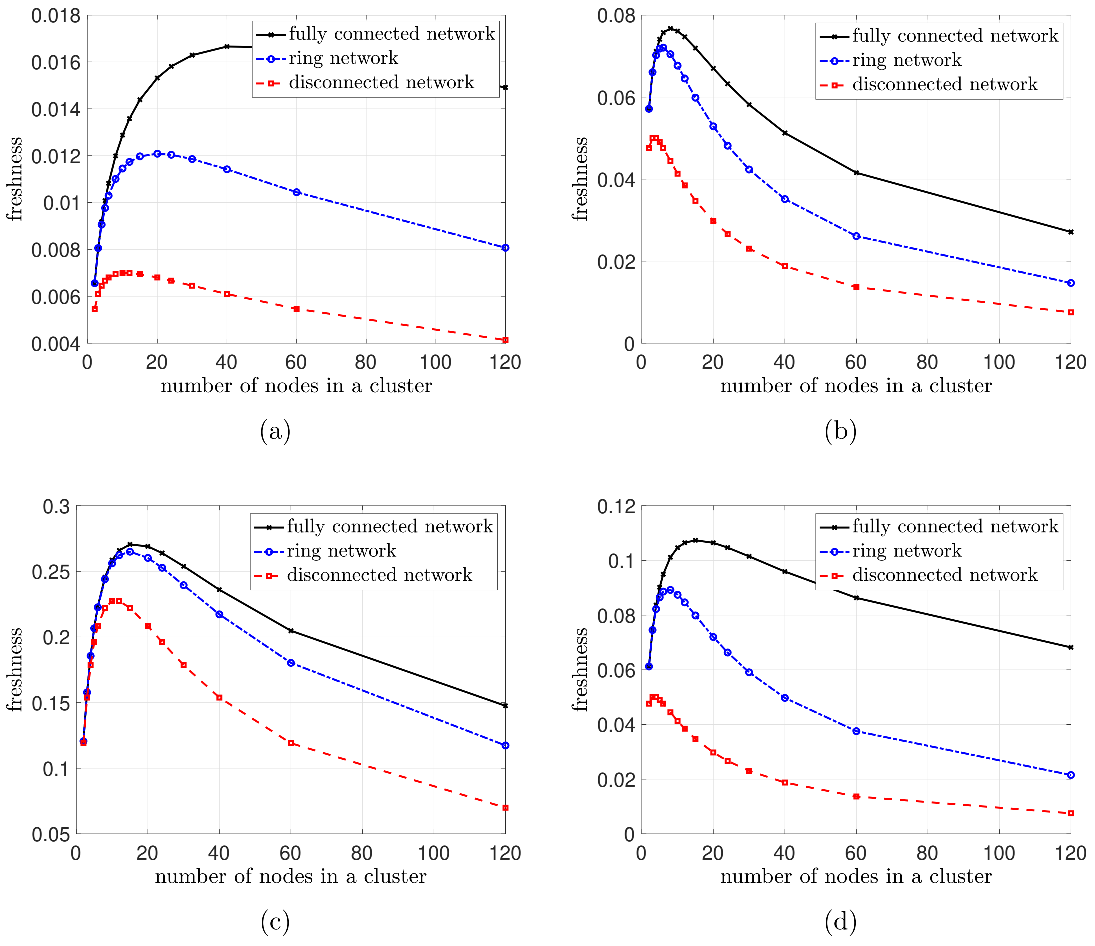


Figure 8.5: Binary freshness of a node with disconnected, ring and fully connected clusters with $n = 120$, $\lambda_e = 1$, (a) $\lambda_s = 1$, $\lambda_c = 1$, $\lambda = 1$, (b) $\lambda_s = 10$, $\lambda_c = 1$, $\lambda = 1$, (c) $\lambda_s = 10$, $\lambda_c = 10$, $\lambda = 1$, (d) $\lambda_s = 10$, $\lambda_c = 1$, $\lambda = 2$.

is a sweet spot where the freshness is maximized for each cluster topology. Since we increase λ_s compared to the previous example, the source can support more cluster heads and the optimal cluster sizes become $k^* = 8$ for fully connected clusters, $k^* = 6$ for ring clusters, $k^* = 3$ or $k^* = 4$ for disconnected clusters.

Third, we consider $\lambda_s = 10$, $\lambda_c = 10$, and $\lambda = 1$, i.e., we increase the cluster heads' update rate compared to the second example. Since we increase the update rate of the cluster heads, the optimal cluster size increases from $k^* = 8$ to $k^* = 15$

for fully connected clusters; from $k^* = 6$ to $k^* = 15$ for ring clusters; and from $k^* = 3$ or $k^* = 4$ to $k^* = 10$ or $k^* = 12$ for disconnected clusters as shown in Fig. 8.5(c).

Fourth, we take $\lambda_s = 10$, $\lambda_c = 1$, and $\lambda = 2$, i.e., compared to the second example, we increase the update rate among the end-nodes. We see in Fig. 8.5(d) that since there is no updates among the nodes in the disconnected clustered network, the optimal cluster size stays the same as $k^* = 3$ or $k^* = 4$. On the other hand, since the update rates among the nodes are increased, the optimal cluster sizes become to $k^* = 15$ for fully connected clusters, and $k^* = 8$ for ring clusters.

8.7 Conclusion

In this chapter, by using the SHS method, we developed recursive formulas to find the binary freshness for a given node in arbitrarily connected networks. Next, we considered structured large gossip networks. We showed that the binary freshness decreases to 0 with rate n^{-1} both for disconnected and ring networks, but with a strictly slower rate in ring networks. We also showed that binary freshness decreases to 0 with much slower rates for fully connected networks; in particular, the form of the freshness function changes from n^{-1} to $n^{-\rho}$ when we go to fully connected networks when the update rate of the source is sufficiently large. We also found recursive expressions for the binary freshness in clustered gossip networks. Via numerical results, we studied the effects of the update rates of the source, cluster heads and the end-nodes on the optimum cluster size.

CHAPTER 9

Timely Tracking of Multiple Counting Random Processes: Tracking Citation Indices of Researchers

9.1 Introduction

In this chapter, we consider the problem of real-time timely estimation of signals, which have different change rates and importance factors, with the goal of finding the optimal individual sampling rates, under a total system sampling rate constraint. Here, we specialize this broader goal to the setting of counting processes and to the context of tracking citation counts of researchers. As abstracted out in Fig. 9.1, Google crawls the web to find and index various items such as documents, images, videos, etc. Focusing on scientific documents, Google Scholar further examines the contents of these documents to extract out citation counts for indexed papers. Google Scholar then needs to update citation counts of individual researchers, which there are many. We model the citation count of each individual researcher as a counting process with a fixed mean, e.g., λ_i for researcher i. Assuming that Google Scholar is resource-constrained, i.e., that it cannot update all researchers all the time, how should it prioritize updating researchers? If it can update only a fraction

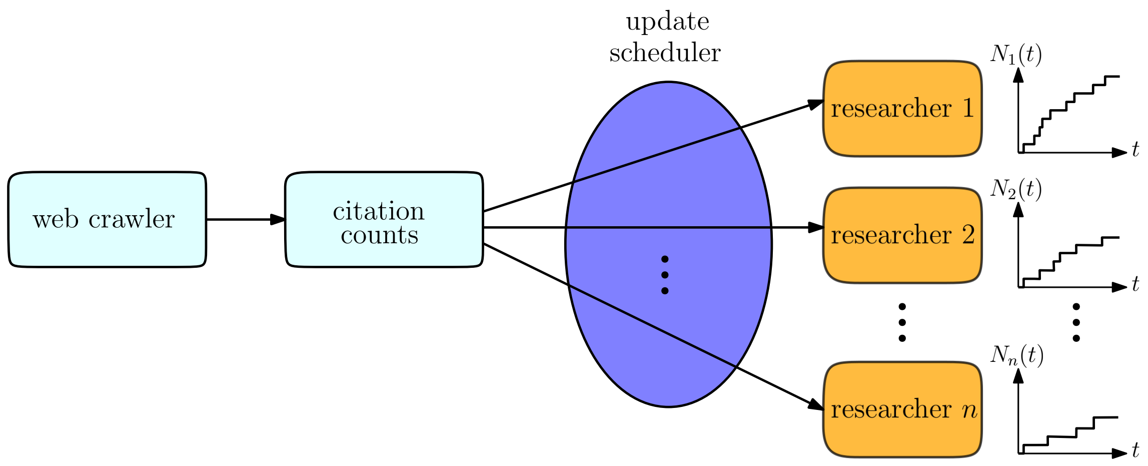


Figure 9.1: Web crawler finds and indexes scientific documents, from which citation counts are extracted upon examining their contents. Scheduler schedules updating citation counts of individual researchers based on their mean citations, and optionally, importance factors, subject to a total update rate.

of all researchers, who should it update? Should it update researchers with higher mean citation rates more often as their citation counts are subject to larger change per unit time? Or should it update researchers with lower mean citation rates more often in order to capture rarer more informative changes?

We model the citation count of each researcher as a counting process with a given mean. In particular, we model citation arrivals for researcher i as a Poisson counting process with rate λ_i. Optionally, we may further assign an importance factor to each researcher, based on their research field or citation count, but this is optional, and does not affect the structure of the results. If an importance coefficient is used, we denote it with μ_i for researcher i. Ideally the updater should update all researchers all the time, however, due to computational limitations, this may not be possible. We model the updater as a resource-constrained entity which has a total update capacity of c, which it should distribute among all researchers. We allocate an update rate ρ_i for updating researcher i. These ρ_i are collectively subject to the total system update capacity of c. We consider the cases of Poisson updates (i.e., updates with exponential inter-update times), deterministic updates, and synchronized updates. We determine the optimal update rates ρ_i subject to the total update rate c in a way to maximize the system freshness.

References that are most closely related to our work are [28] and [30]. Reference [28] considers the problem of finding optimal crawl rates to keep the information in a search engine fresh while maintaining the constraints on crawling rates imposed by the websites and also the total crawl rate constraint of the search engine, in the presence of non-uniform importance scores and change rates for the websites. Reference [30] focuses on remote real-time reconstruction of a single Poisson counting process using uniform sampling. While taking more samples helps reconstruct the signal better, this increases queuing delays, which inherently affects the real-time signal estimation negatively. Reference [30] studies this trade-off and finds the optimal sampling rate. Our timeliness metric is similar to the one considered in [30], which is the difference of a counting process and its sampled (updated, in our case) version. However, we consider multiple counting processes with different arrival rates and importance factors, and optimize our update rates for all processes jointly under a total update rate constraint. Similar to [28], we consider exponential arrival and sampling times in a constrained manner, and allow for importance coefficients, however, our timeliness metric and our overall problem setting are different.

In this chapter, we first find an analytical expression for the long-term average difference between the actual and updated counting processes. We then minimize this expression as a function of the update rates of individual researchers subject to the overall update rate. We show that the optimal update rates are proportional to the *square roots* of mean citation rates of the researchers for constant importance factors. Thus, it is optimal to update more prolific researchers more often, however, the proportionality is sub-linear and in the form of square root of the mean citation rate, i.e., there are diminishing returns due to the concavity (sub-linearity) of the square root function. We show that if the importance factors of the researchers are linear in their mean citation rates, then the optimal update rates are linear in their mean citation rates as well. We finally remark that other *square root* results have appeared in completely different settings in caching problems. In particular, it was found in [93] that, in order to keep the files fresh in a caching system, the (uniform) update rates of the files should be chosen proportional to the square roots of their popularity indices. In addition, it was shown in [157] that, in order to minimize the average number of websites searched, the number of websites that a file is mirrored should be chosen proportional to the square root of the file's popularity.

9.2 System Model and Problem Formulation

Consider n researchers. Let $N_i(t)$ denote the number of citations of researcher i. We model $N_i(t)$ as a Poisson process with rate λ_i, and assume that $N_i(t)$, for $i = 1, \ldots, n$, are independent. Let $t_{i,j}$ denote the time instance when the updater updates the number of citations of researcher i for the jth time. We denote the inter-update time between the jth and $(j-1)$th updates for researcher i as $\tau_{i,j}$. Based on these samples, the updater generates a real-time estimate of the counting process $N_i(t)$

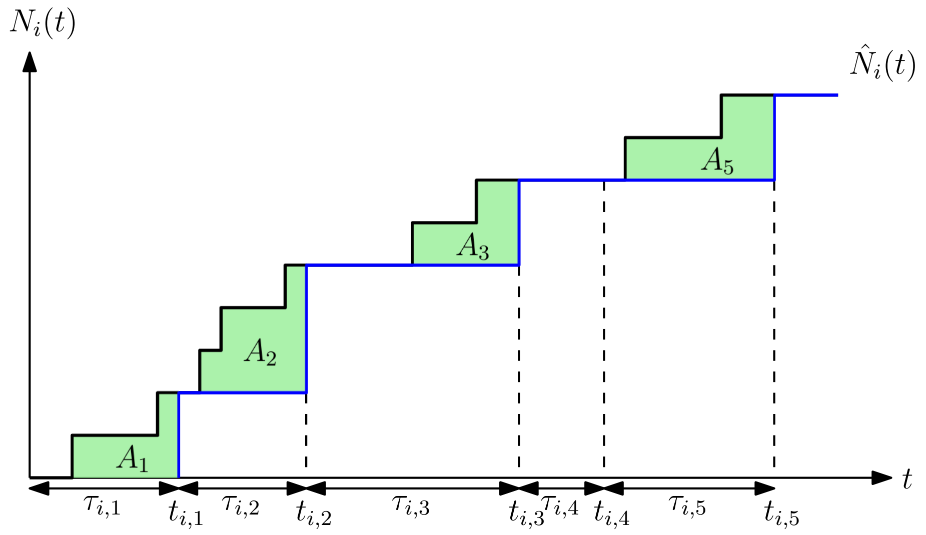


Figure 9.2: The number of citations, $N_i(t)$, and the estimated number of citations, $\hat{N}_i(t)$, for researcher i. A_j denotes the total estimation error in $[t_{i,j-1}, t_{i,j})$.

as $\hat{N}_i(t)$, where

$$\hat{N}_i(t) = N_i(t_{i,j-1}), \quad t_{i,j-1} \leq t < t_{i,j}. \tag{9.1}$$

Fig. 9.2 shows $N_i(t)$ and $\hat{N}_i(t)$ with black and blue lines.

Similar to [30], we use the average difference between the actual and updated processes as a measure of timeliness,

$$\Delta_i(T) = \frac{1}{T} \int_0^T \left(N_i(t) - \hat{N}_i(t) \right) dt. \tag{9.2}$$

If there are m updates in the interval $[0, T]$, then $\Delta_i(T)$ is

$$\Delta_i(T) = \frac{1}{T} \sum_{j=1}^{m} A_j, \tag{9.3}$$

where A_j is the difference in the interval $[t_{i,j-1}, t_{i,j})$, see Fig. 9.2. Then, the long term average difference for researcher i is $\Delta_i = \lim_{T\to\infty} \Delta_i(T)$, and can be written as [10], [1]

$$\Delta_i = \lim_{T\to\infty} \frac{1}{T}\sum_{j=1}^{m} A_j = \lim_{T\to\infty} \frac{m}{T}\cdot\frac{1}{m}\sum_{j=1}^{m} A_j = \rho_i \mathbb{E}[A]. \tag{9.4}$$

Similar to the derivation in [30], conditioned on an arbitrary jth inter-update time, i.e., $\tau_i = \tau_{i,j} = d$, and the number of citation arrivals in that time interval $[t_{i,j-1}, t_{i,j-1}+d)$, i.e., $\tilde{N}_i(d) = N_i(t_{i,j-1}+d) - N_i(t_{i,j-1}) = k$, the expected difference is

$$\mathbb{E}\left[A|\tilde{N}_i(d) = k, \tau_i = d\right] = \frac{kd}{2}. \tag{9.5}$$

Thus, we have

$$\mathbb{E}\left[A|\tau_i = d\right] = \mathbb{E}\left[\mathbb{E}\left[A|\tilde{N}_i(d) = k, \tau_i = d\right]\right] = \frac{\lambda_i d^2}{2}. \tag{9.6}$$

In the following subsections, we present three different models for updating the citation numbers.

9.2.1 Model 1: Poisson Updater

In this model, shown in Fig. 9.3, the inter-update times for researcher i are exponential with rate ρ_i. Update processes for different researchers are independent.

[1]The existence of this limit is shown in [158, Lemmas 1 and 2] for the deterministic updater model, and in [158, Lemmas 5 and 6] for the Poisson updater model considered in this work.

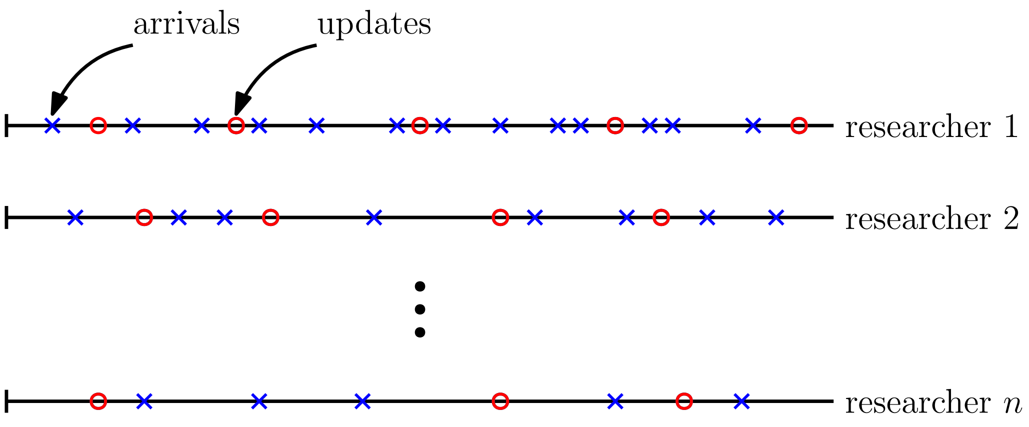


Figure 9.3: Poisson updater: Inter-update times are exponential with rate ρ_i.

Continuing from (9.6), we find $\mathbb{E}[A]$ using exponential distribution as,

$$\mathbb{E}[A] = \int_0^\infty \mathbb{E}\left[A|\tau_i = t\right] f_{\tau_i}(t)dt = \frac{\lambda_i}{2}\mathbb{E}\left[\tau_i^2\right] = \frac{\lambda_i}{\rho_i^2}. \tag{9.7}$$

Thus, the long term average difference Δ_i in (9.4) with a Poisson updater is

$$\Delta_i = \frac{\lambda_i}{\rho_i}. \tag{9.8}$$

9.2.2 Model 2: Deterministic Updater

In this model, shown in Fig. 9.4, the inter-update times are deterministic and chosen optimally. Similar to [93], given that there are m_i updates for researcher i in the time interval $[0, T]$, the optimal inter-update times should be chosen equal to each other, i.e., $\tau_{i,j} = \frac{T}{m_i+1}$, for all j. Letting $T \to \infty$, this update scheme results in uniform sampling with rate ρ_i for researcher i where $\rho_i = \lim_{T\to\infty} \frac{m_i}{T}$. By using $d_i = \frac{1}{\rho_i}$ and (9.6), we obtain $\mathbb{E}[A] = \frac{\lambda_i}{2\rho_i^2}$. Thus, the long term average difference Δ_i

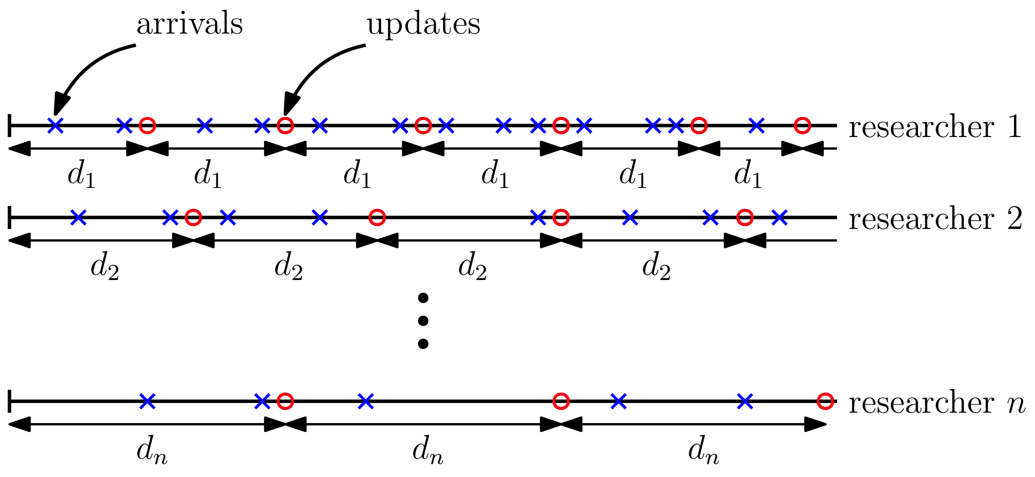


Figure 9.4: Deterministic updater: Inter-update times are equal with $d_i = \frac{1}{\rho_i}$.

in (9.4) with a deterministic (and uniform) updater is

$$\Delta_i = \frac{\lambda_i}{2\rho_i}. \tag{9.9}$$

9.2.3 Model 3: Common Synchronized Probabilistic Updater

In this model, shown in Fig. 9.5, the updater has a common synchronized update schedule that applies to all researchers. The inter-update times of the common updater are exponential with rate ρ. At each update instant, researcher i is updated with probability p_i independently of other researchers. Thus, inter-update times for researcher i are exponential with rate ρp_i. Note that here, we create the Poisson updates for researcher i by thinning the Poisson common updates using probabilistic updates according to p_i. The main problem, therefore, is to choose p_i for each researcher as it determines its mean update rate. This problem is the same as the one in Section 9.2.1 and in the optimal policy p_i is $p_i = \frac{\rho_i}{\rho}$ assuming ρ is sufficiently large to have feasible p_i, i.e., $0 \leq p_i \leq 1$, for all i.

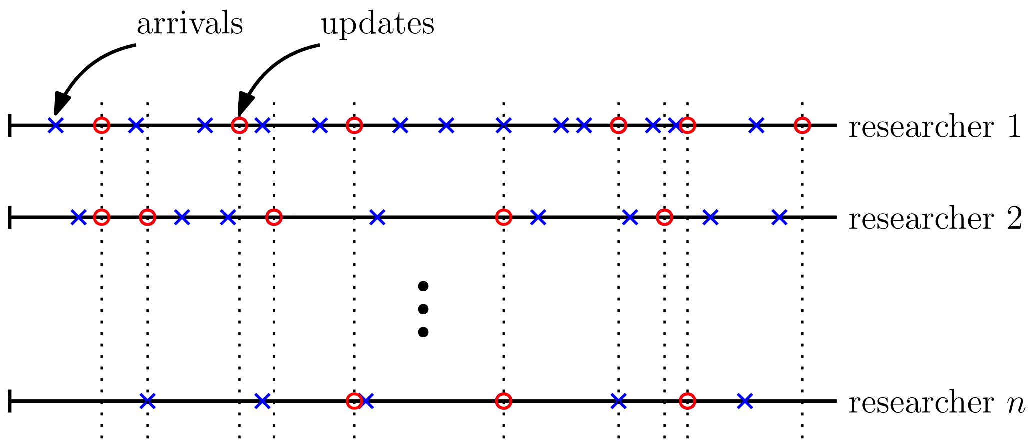


Figure 9.5: Common synchronized updater: Common synchronized inter-update times are exponential with rate ρ. At each common update opportunity, researcher i is updated with probability p_i

.

9.2.4 Problem Formulation

Researcher i has the mean citation rate λ_i. In addition, and optionally, we consider an importance factor, μ_i, for researcher i. This may be removed by choosing all $\mu_i = \mu$. Then, the total long term average difference (over all researchers) becomes $\Delta = \sum_{i=1}^{n} \mu_i \Delta_i$, where per researcher difference, Δ_i, is given by (9.8) for the Poisson updater and common synchronized updater models, and by (9.9) for the deterministic updater model. The expressions in (9.8) and (9.9) differ only by a factor of 2, which is inconsequential for optimization purposes. Therefore, without loss of generality, from now on, we use the expression in (9.8). In addition, due to computational limitations, the updater is subject to a total update rate constraint $\sum_{i=1}^{n} \rho_i \leq c$. Our aim is to find the optimal update rates for all researchers, ρ_i, for $i = 1, \ldots, n$, such that the total long term average difference, Δ, is minimized while satisfying the constraint on the total update rate. Thus, our optimization problem is,

$$\begin{aligned}
\min_{\{\rho_i\}} \quad & \sum_{i=1}^{n} \frac{\mu_i \lambda_i}{\rho_i} \\
\text{s.t.} \quad & \sum_{i=1}^{n} \rho_i \leq c \\
& \rho_i \geq 0, \quad i = 1, \ldots, n.
\end{aligned} \tag{9.10}$$

We solve the optimization problem in (9.10) in the next section.

9.3 The Optimal Solution

The optimization problem in (9.10) is convex as the cost function is convex and the constraints are linear. We introduce the Lagrangian function [149] for (9.10) as

$$\mathcal{L} = \sum_{i=1}^{n} \frac{\mu_i \lambda_i}{\rho_i} + \beta \left(\sum_{i=1}^{n} \rho_i - c \right) - \sum_{i=1}^{n} \nu_i \rho_i, \tag{9.11}$$

where $\beta \geq 0$ and $\nu_i \geq 0$ for all i. Next, we write the KKT conditions as

$$\frac{\partial \mathcal{L}}{\partial \rho_i} = -\frac{\mu_i \lambda_i}{\rho_i^2} + \beta - \nu_i = 0, \tag{9.12}$$

for all i, and the complementary slackness conditions as

$$\beta \left(\sum_{i=1}^{n} \rho_i - c \right) = 0, \tag{9.13}$$

$$\nu_i \rho_i = 0, \tag{9.14}$$

for all i. Since the optimization problem in (9.10) is convex, the KKT conditions are necessary and sufficient.

First, we observe that the total update rate constraint $\sum_{i=1}^{n} \rho_i \leq c$ must be satisfied with equality. If there is an update rate allocation policy such that $\sum_{i=1}^{n} \rho_i < c$, then we can achieve a lower average difference by increasing any ρ_i as the cost function of (9.10) is a decreasing function of ρ_i. Thus, in the optimal update rate allocation policy, we must have $\sum_{i=1}^{n} \rho_i = c$ and $\beta \geq 0$ due to (9.13).

Next, we note that in the optimal policy, we must have $\rho_i > 0$, for all i, as $\rho_i = 0$ leads to infinite objective function in (9.10) which clearly cannot be an optimal solution. Thus, for the optimal rate allocations, we have $\rho_i > 0$ and $\nu_i = 0$, for all i, due to (9.14).

From (9.12), we find $\rho_i = \sqrt{\frac{\mu_i \lambda_i}{\beta}}$. By using $\sum_{i=1}^{n} \rho_i = c$, we solve $\beta = \frac{\left(\sum_{i=1}^{n} \sqrt{\mu_i \lambda_i}\right)^2}{c^2}$, which gives the optimal policy,

$$\rho_i = \frac{c\sqrt{\mu_i \lambda_i}}{\left(\sum_{j=1}^{n} \sqrt{\mu_j \lambda_j}\right)}, \quad i = 1, \ldots, n. \tag{9.15}$$

Using the optimal rate allocation policy in (9.15), we obtain

$$\Delta_i = \frac{\sqrt{\lambda_i}\left(\sum_{j=1}^{n} \sqrt{\mu_j \lambda_j}\right)}{c\sqrt{\mu_i}}, \quad i = 1, \ldots, n, \tag{9.16}$$

and the total long term average difference Δ as

$$\Delta = \frac{\left(\sum_{j=1}^{n} \sqrt{\mu_j \lambda_j}\right)^2}{c}. \tag{9.17}$$

Thus, the optimal update rates allocated to researchers in (9.15) are proportional to the square roots of their importance factors, μ_i, multiplied by their mean citation rates, λ_i. We note that if we ignore the importance factors, i.e., $\mu_i = \mu = 1$, then the optimal update rates are proportional to the square roots of the mean citation rates. On the other hand, if we choose the importance factors as proportional to the mean citation rates, i.e., $\mu_i = \alpha\lambda_i$, then the optimal update rates become linear in the mean citation rates.

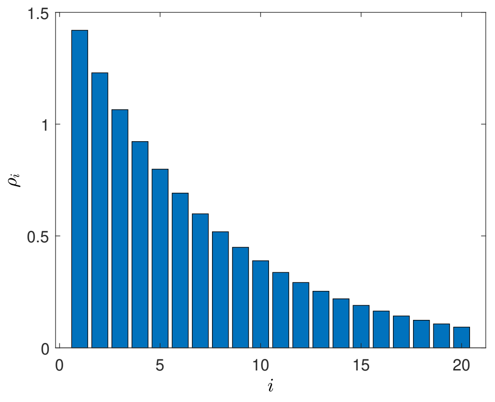
9.4 Numerical Results

In this section, we provide three numerical results. In the first two examples, we choose the mean citation rates as

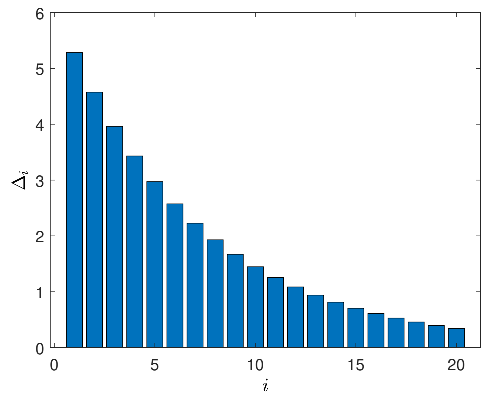
$$\lambda_i = ar^i, \quad i = 1, \dots, n, \tag{9.18}$$

where $a > 0$ and $0 < r \leq 1$.

In the first example, we take $a = 10$, $r = 0.75$, $n = 20$ and $c = 10$. For this example, we use uniform importance coefficients, i.e., $\mu_i = 1$, for all i. We observe in Fig. 9.6(b) that researchers with higher mean citation rates have higher long term average difference Δ_i even though they are updated with higher update rates shown in Fig. 9.6(a). Further, we observe in Fig. 9.6(a) that due to diminishing returns caused by the square root allocation policy, update rates of the researchers with low mean citation rates are still comparable to the update rates of the researchers with high mean citation rates.



(a)



(b)

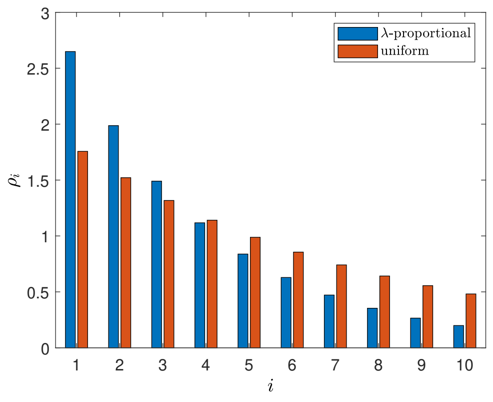
Figure 9.6: (a) Optimal update rate allocation for each researcher, and (b) the corresponding optimal long term average difference Δ_i, when we use uniform importance coefficients $\mu_i = 1$, with λ_i given in (9.18), with $a = 10$ and $r = 0.75$ for $n = 20$.

In the second example, we consider the case where the importance factors are chosen proportional to the mean citation rates of the researchers. We call such coefficients as λ-*proportional* importance coefficients, which are given by

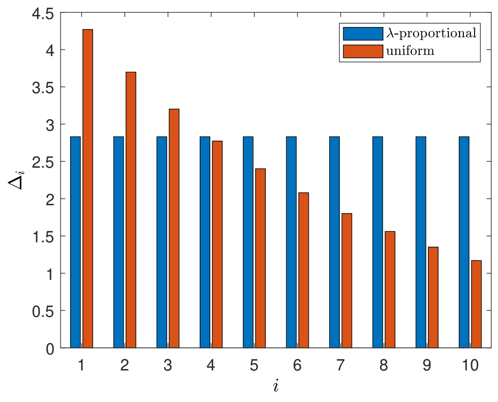
$$\mu_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}, \quad i = 1, \ldots, n. \tag{9.19}$$

In order to make a fair comparison between the λ-*proportional* and *uniform* importance coefficients, we scale the uniform importance coefficients as $\mu_i = \frac{1}{n}$, for all i.

As mentioned at the end of Section 9.3, when we use λ-*proportional* importance coefficients, the optimal update rates become linear in the mean citation rates. We observe in Fig. 9.7(a) that using λ-*proportional* importance coefficients favor researchers with higher mean citation rates as their update rates increase compared to the update rates with the uniform importance coefficients. Further, we observe



(a)



(b)

Figure 9.7: (a) Optimal update rate allocation for each researcher, and (b) the corresponding optimal long term average difference Δ_i, when we use λ-*proportional* and uniform importance coefficients, with λ_i given in (9.18), with $a = 10$ and $r = 0.75$ for $n = 10$.

in Fig. 9.7(b) that the long term average differences are equal to each other when λ-*proportional* importance coefficients are used.

In the third example, we choose the mean citation rates as

$$\lambda_i = \frac{ar^i}{\sum_{j=1}^{n} r^j}, \quad i = 1, \ldots, n, \tag{9.20}$$

which satisfy $\sum_{i=1}^{n} \lambda_i = a$. Note that, by this selection, we force total citation means of all researchers to be a constant. For this example, we take $n = 10$, $a = 1$ and consider three different r, which are $r = 0.5, 0.75, 1$. Note also that, a smaller r corresponds to a less even (more polarized) distribution of total mean citation rates among the researchers. We use uniform importance coefficients and plot achieved Δ with respect to c in Fig. 9.8. We observe in Fig. 9.8 that more polarized distribution of mean citation rates (smaller r) yields a lower Δ for the

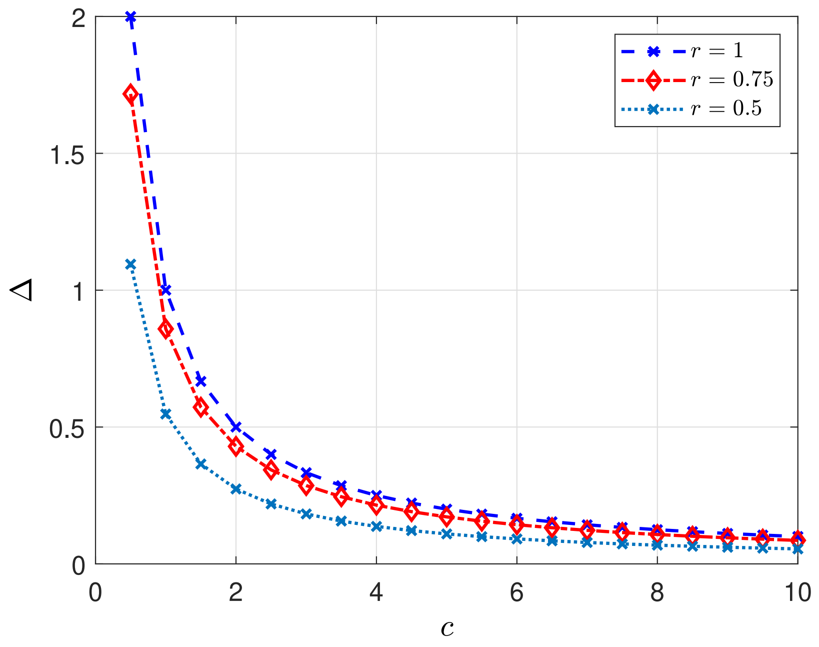


Figure 9.8: Total long term average difference Δ with respect to c, when uniform importance coefficients are used and λ_i are given in (9.20), with $a = 1$ and $r = 0.5, 0.75, 1$ for $n = 10$.

system, as we exploit the differences among the researchers by allocating even higher update rates to researchers with higher mean citation rates. As an aside, we note that if we used λ-*proportional* importance coefficients, we would have a Δ which is independent of individual λ_i that depends only on the sum of λ_i which is a here. This achieved Δ is also equal to the Δ achieved with uniform importance coefficients when $r = 1$ which is shown as the blue dashed line in Fig. 9.8. Thus, if we use λ-*proportional* importance coefficients, the achieved Δ is independent of the mean citation rate distribution among the researchers, but it results in higher Δ. In other words, uniform importance coefficients achieve lower Δ compared to λ-*proportional* importance coefficients in this case for $r < 1$.

9.5 Conclusion

In this chapter, we considered the problem of timely updating of citation counts by a resource-constrained updater. We showed that the optimal policy is to choose the update rates of individual researchers proportional to the square roots of their mean citation rates multiplied by their importance factors (if any).

Next, we discuss limitations of our model. First, we note that we modeled the citation numbers of a researcher as a counting process, which is monotonically increasing and increments one at a time. The monotonically increasing nature of the counting process assumes that published articles are always available and cannot be withdrawn or changed, which may not be the case, especially if the crawler cannot reach certain websites that it used to reach, and lose access to previously counted publications. In addition, one at a time increments assume that citations come one by one, which may not be the case as conferences publish all the articles simultaneously in proceedings, and journals publish articles in monthly issues. That is, the increments in citations for each researcher may be more than one at a time. Thus, modeling the number of citations with a simple Poisson process may not be sufficient. However, as the researchers increasingly publish their works as they are completed on their personal websites or academic websites such as arXiv, it may still be acceptable to model arrivals as Poisson processes. Even then though, multiple citations to the same researcher from a single article will not be captured by the model in this chapter. In addition, considering the fact that researchers often collaborate and publish articles jointly, the arrivals of citations for different researchers might be correlated. In this chapter, we considered the case where the citation arrivals for researchers are independent counting processes. This may be feasible by focusing on researchers with no common publications.

CHAPTER 10

Timely Tracking of Multiple Binary Random Processes: Tracking Infection Status of Individuals in a Population

10.1 Introduction

In this chapter, we consider the problem of real-time timely tracking of multiple binary random processes via a resource constrained Poisson updater. As a particular case, we consider the problem of timely tracking of an infectious disease, e.g., covid-19, in a population of n people. In this problem, a health care provider wants to detect infected people as quickly as possible in order to take precautions such as isolating them from the rest of the population. The health care provider also wants to detect people who recovered from the disease as soon as possible since these people need to return to work which is especially critical in sectors such as health care, food retail, and public transportation. Ideally, the health care provider should test all people all the time. However, as the total test rate is limited, the question is how frequently the health care provider should apply tests on these people when their infection and recovery rates are known. In a broader sense, this problem is related to timely tracking of multiple processes in a resource-constrained setting

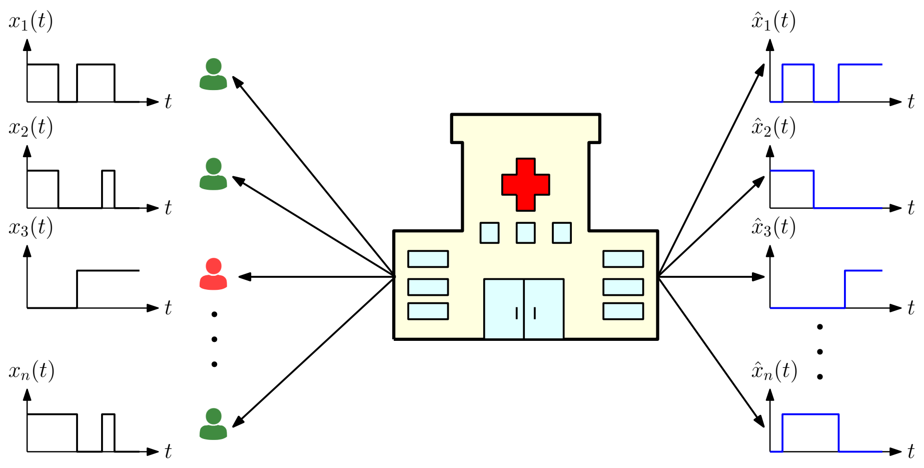


Figure 10.1: System model. There are n people whose infection status are given by $x_i(t)$. The health care provider applies tests on these people. Based on the test results, estimations for the infection status $\hat{x}_i(t)$ are generated. Infected people are shown in red color and healthy people are shown in green color.

where each process takes binary values of 0 and 1 with different change rates.

Recent studies have shown that people who recovered from infectious diseases such as covid-19 can be reinfected. Furthermore, the recovery times of individuals from the disease may vary significantly. For these reasons, in this problem, the ith person gets infected with rate λ_i which is independent of the others. Similarly, the ith person recovers from the disease with rate μ_i.[1] We denote the infection status of the ith person as $x_i(t)$ (shown with the black curves on the left in Fig. 10.1) which takes the value 1 when the person is infected and the value 0 when the person is healthy. The health care provider applies tests to people marked as healthy with rate s_i and to people marked as infected with rate c_i. Based on the test results, the health care provider forms an estimate for the infection status of the ith person

[1]We note that the index i may represent a specific individual or a group of individuals that have common features such as age, gender, profession. For example, $i = 1$ may denote men between ages 70-75 who live in nursing homes, and $i = 2$ may denote women between ages of 20-25 who work in the medical field, and so on. Therefore, depending on the demographics, coefficients λ_i and μ_i may be statistically known by the health care provider.

denoted by $\hat{x}_i(t)$ (shown with the blue curves on the right in Fig. 10.1) which takes the value 1 when the most recent test result is positive and the value 0, otherwise.

We measure the timeliness of the tracking process by the difference between the actual infection status of people and the real-time estimate of the health care provider which is based on the most recent test results. We note that the difference can occur in two different cases: i) when the person is sick ($x_i(t) = 1$) and the health care provider maps this person as healthy ($\hat{x}_i(t) = 0$), and ii) when the person recovers from the disease ($x_i(t) = 0$) but the health care provider still considers this person as infected ($\hat{x}_i(t) = 1$). The former case represents the error due to late detection of infected people, while the latter case represents the error due to late detection of healed people. Depending on the health care provider's preferences, detecting infected people may be more important than detecting recovered people, or vice versa.

Most relevant to our work, the real-time timely estimation of a single and multiple counting processes [30, 36], a Wiener process [31], a random walk process [118], a binary Markov source [34] have been studied. The work that is closest to our work is reference [34] where the remote estimation of a symmetric binary Markov source is studied in a time-slotted system by finding the optimal sampling policies via formulating a Markov Decision Process (MDP) for real-time error, AoI and AoII metrics. Different from [34], in our work, we consider real-time timely estimation of multiple non-symmetric binary sources for a continuous time system. We note that in our work, the sampler (the health care provider) does not know the states of the sources (infection status of people), and thus takes the samples (applies medical tests) randomly with fixed rates. Thus, in our work, we optimize the test rates of people to minimize the real-time estimation error.

In this chapter, we consider the real-time timely tracking of infection status of n people. We first find an analytical expression for the long-term average difference between the actual infection status of people and the estimate of the health care provider based on test results. Then, we propose an alternating minimization based algorithm to find the test rates s_i and c_i for all people. We observe that if the total test rate is limited, we may not apply tests on all people equally. Increasing the total test rate helps track the infection status of people better, and increasing the size of the population increases diversity which may be exploited to improve the performance. Finally, depending on the health care provider's priorities, we can allocate more tests to people marked as healthy to detect the infections more quickly or to people marked as infected to detect the recoveries more quickly.

10.2 System Model

We consider a population of n people. We denote the infection status of the ith person at time t as $x_i(t)$ (black curve in Fig. 10.2(a)) which takes binary values 0 or 1 as follows,

$$x_i(t) = \begin{cases} 1, & \text{if the } i\text{th person is infected at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{10.1}$$

In this chapter, we consider a model where each person can be infected multiple times after recovering from the disease. We denote the time interval that the ith person stays healthy for the jth time as $W_i(j)$ which is exponentially distributed with rate λ_i. We denote the recovery time for the ith person after infected with the virus for the jth time as $R_i(j)$ which is exponentially distributed with rate μ_i.

A health care provider wants to track the infection status of each person. Based on the test results at times $t_{i,\ell}$, the health care provider generates an estimate for the status of the ith person denoted as $\hat{x}_i(t)$ (blue curve in Fig. 10.2(a)) by

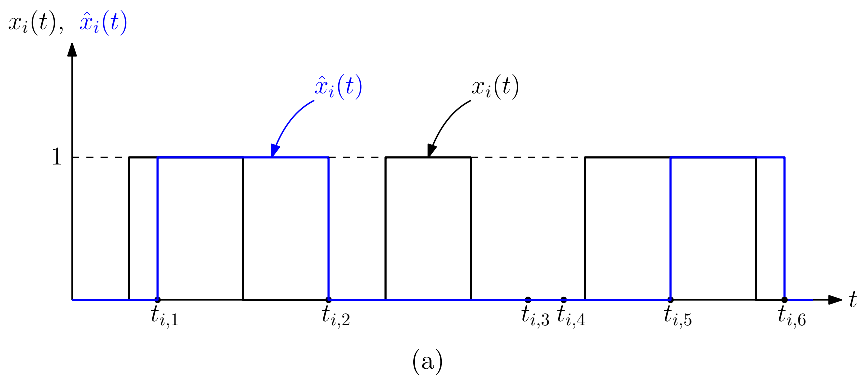
$$\hat{x}_i(t) = x_i(t_{i,\ell}), \quad t_{i,\ell} \leq t < t_{i,\ell+1}. \tag{10.2}$$

When $\hat{x}_i(t)$ is 1, the health care provider applies the next test to the ith person after an exponentially distributed time with rate c_i. When $\hat{x}_i(t)$ is 0, the next test is applied to the ith person after an exponentially distributed time with rate s_i.

An estimation error happens when the actual infection status of the ith person, $x_i(t)$, is different than the estimate of the health care provider, $\hat{x}_i(t)$, at time t. This could happen in two ways: when $x_i(t) = 1$ and $\hat{x}_i(t) = 0$, i.e., when the ith person is sick, but it has not been detected by the health care provider, and when $x_i(t) = 0$ and $\hat{x}_i(t) = 1$, i.e., when the ith person has recovered, but the health care provider does not know that the ith person has recovered.

We denote the error caused by the former case, i.e., when $x_i(t) = 1$ and $\hat{x}_i(t) = 0$, by $\Delta_{i1}(t)$ (green areas in Fig. 10.2(b)),

$$\Delta_{i1}(t) = \max\{x_i(t) - \hat{x}_i(t), 0\}, \tag{10.3}$$



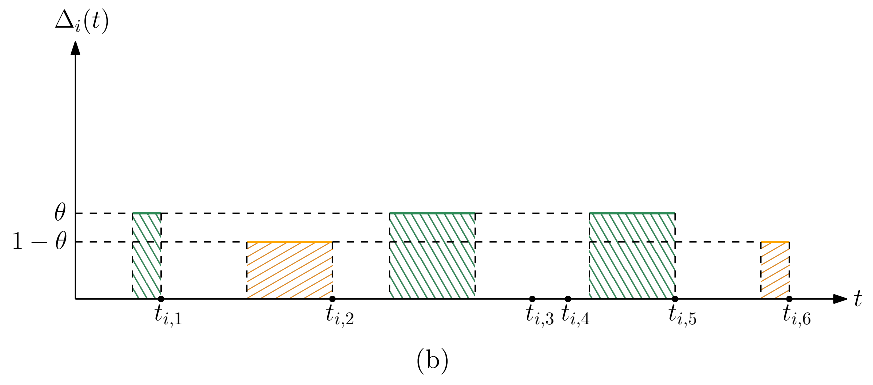


Figure 10.2: (a) A sample evolution of $x_i(t)$ and $\hat{x}_i(t)$, and (b) the corresponding $\Delta_i(t)$ in (10.5). Green areas correspond to the error caused by $\Delta_{i1}(t)$ in (10.3). Orange areas correspond to the error caused by $\Delta_{i2}(t)$ in (10.4).

and we denote the error caused by the latter case, i.e., when $x_i(t) = 0$ and $\hat{x}_i(t) = 1$, by $\Delta_{i2}(t)$ (orange areas in Fig. 10.2(b)),

$$\Delta_{i2}(t) = \max\{\hat{x}_i(t) - x_i(t), 0\}. \tag{10.4}$$

Then, the total estimation error for the ith person $\Delta_i(t)$ is

$$\Delta_i(t) = \theta\Delta_{i1}(t) + (1-\theta)\Delta_{i2}(t), \tag{10.5}$$

where θ is the importance factor in $[0, 1]$. A large θ gives more importance to the detection of infected people, and a small θ gives more importance to the detection of recovered people.

We define the long-term weighted average difference between $x_i(t)$ and $\hat{x}_i(t)$ as

$$\Delta_i = \lim_{T\to\infty} \frac{1}{T} \int_0^T \Delta_i(t)dt. \tag{10.6}$$

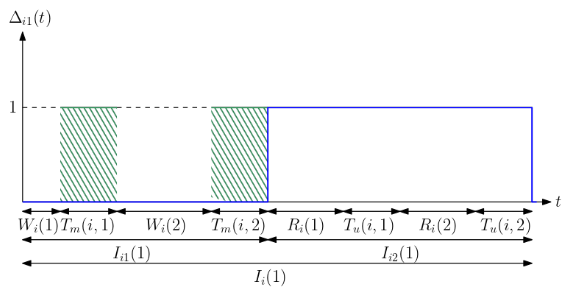
Then, the overall average difference of all people Δ is

$$\Delta = \frac{1}{n} \sum_{i=1}^{n} \Delta_i. \tag{10.7}$$

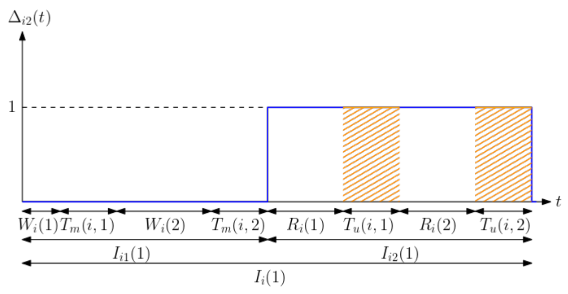
Our aim is to track the infection status of all people. Due to limited resources, there is a total test rate constraint $\sum_{i=1}^{n} s_i + \sum_{i=1}^{n} c_i \leq C$. Thus, our aim is to find the optimal test rates s_i and c_i to minimize Δ in (10.7) while satisfying this total test rate constraint. We formulate the following problem,

$$\begin{aligned} \min_{\{s_i,c_i\}} \quad & \Delta \\ \text{s.t.} \quad & \sum_{i=1}^{n} s_i + \sum_{i=1}^{n} c_i \leq C \\ & s_i \geq 0, \quad c_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{10.8}$$

In the next section, we find the total average difference Δ.



(a)



(b)

Figure 10.3: A sample evolution of (a) $\Delta_{i1}(t)$, and (b) $\Delta_{i2}(t)$ in a typical cycle.

10.3 Average Difference Analysis

We first find analytical expressions for $\Delta_{i1}(t)$ in (10.3) and $\Delta_{i2}(t)$ in (10.4). We note that $\Delta_{i1}(t)$ can be equal to 1 when $\hat{x}_i(t) = 0$ and is always equal to 0 when $\hat{x}_i(t) = 1$. Assume that at time 0, both $x_i(0)$ and $\hat{x}_i(0)$ are 0. After an exponentially distributed time with rate λ_i, which is denoted by W_i, the ith person is infected, and thus $x_i(t)$ becomes 1. At that time, since $\hat{x}_i(t) = 0$, $\Delta_{i1}(t)$ becomes 1. $\Delta_{i1}(t)$ will be equal to 0 again either when the ith person recovers from the disease which happens after R_i which is exponentially distributed with rate μ_i or when the health care provider performs a test on the ith person after D_i which is exponentially distributed with rate s_i. We define $T_m(i)$ as the earliest time at which one of these two cases happens, i.e., $T_m(i) = \min\{R_i, D_i\}$. We note that $T_m(i)$ is also exponentially distributed with rate $\mu_i + s_i$, and we have $\mathbb{P}(T_m(i) = R_i) = \frac{\mu_i}{\mu_i+s_i}$ and $\mathbb{P}(T_m(i) = D_i) = \frac{s_i}{\mu_i+s_i}$. If the ith person recovers from the disease before testing, we return to the initial case where both $x_i(t)$ and $\hat{x}_i(t)$ are equal to 0 again. In this case, this cycle repeats itself, i.e., the ith person becomes sick again after W_i and $\Delta_{i1}(t)$ remains as 1 until either the person recovers or the health care provider performs a test which takes another $T_m(i)$ duration. If the health care provider performs a test before the person recovers, then $\hat{x}_i(t)$ becomes 1. We denote the time interval for which $\hat{x}_i(t)$ stays at 0 as I_{i1} which is given by

$$I_{i1} = \sum_{\ell=1}^{K_1} T_m(i,\ell) + W_i(\ell), \tag{10.9}$$

where K_1 is geometric with rate $\mathbb{P}(T_m(i) = D_i) = \frac{s_i}{\mu_i+s_i}$. Due to [147, Prob. 9.4.1], $\sum_{\ell=1}^{K_1} T_m(i,\ell)$ and $\sum_{\ell=1}^{K_1} W_i(\ell)$ are exponentially distributed with rates s_i and $\frac{\lambda_i s_i}{\mu_i+s_i}$, respectively. As $\mathbb{E}[I_{i1}] = \mathbb{E}[\sum_{\ell=1}^{K_1} T_m(i,\ell)] + \mathbb{E}[\sum_{\ell=1}^{K_1} W_i(\ell)]$, we have

$$\mathbb{E}[I_{i1}] = \frac{1}{s_i} + \frac{s_i+\mu_i}{s_i\lambda_i}. \tag{10.10}$$

When $\hat{x}_i(t) = 1$, the health care provider marks the ith person as infected. The ith person recovers from the virus after R_i. After the ith person recovers, either the health care provider performs a test after Z_i which is exponentially distributed with rate c_i or the ith person is reinfected with the virus which takes W_i time. We define $T_u(i)$ as the earliest time at which one of these two cases happens, i.e., $T_u(i) = \min\{W_i, Z_i\}$. Similarly, we note that $T_u(i)$ is exponentially distributed with rate $\lambda_i + c_i$, and we have $\mathbb{P}(T_u(i) = W_i) = \frac{\lambda_i}{\lambda_i+c_i}$ and $\mathbb{P}(T_u(i) = Z_i) = \frac{c_i}{\lambda_i+c_i}$. If the person is reinfected with the virus before a test is applied, this cycle repeats itself, i.e., the ith person recovers after another R_i, and then either a test is applied to the ith person, or the person is infected again which takes another $T_u(i)$. If the health care provider performs a test to the ith person before the person is reinfected, the health care provider marks the ith person as healthy again, i.e., $\hat{x}_i(t)$ becomes 0. We denote the time interval that $\hat{x}_i(t)$ is equal to 1 as I_{i2} which is given by

$$I_{i2} = \sum_{\ell=1}^{K_2} T_u(i,\ell) + R_i(\ell), \tag{10.11}$$

where K_2 is geometric with rate $\mathbb{P}(T_u(i) = Z_i) = \frac{c_i}{\lambda_i + c_i}$. Similarly, $\sum_{\ell=1}^{K_2} T_u(i,\ell)$ and $\sum_{\ell=1}^{K_2} R_i(\ell)$ are exponentially distributed with rates c_i and $\frac{c_i\mu_i}{\lambda_i + c_i}$, respectively. As $\mathbb{E}[I_{i2}] = \mathbb{E}[\sum_{\ell=1}^{K_2} T_u(i,\ell)] + \mathbb{E}[\sum_{\ell=1}^{K_2} R_i(\ell)]$, we have

$$\mathbb{E}[I_{i2}] = \frac{1}{c_i} + \frac{c_i + \lambda_i}{c_i\mu_i}. \tag{10.12}$$

We denote the time interval between the jth and $(j+1)$th times that $\hat{x}_i(t)$ changes from 1 to 0 as the jth cycle $I_i(j)$ where $I_i(j) = I_{i1}(j) + I_{i2}(j)$. We note that $\Delta_{i1}(t)$ is always equal to 0 during $I_{i2}(j)$, i.e., $\hat{x}_i(t) = 1$, and $\Delta_{i1}(t)$ is equal to 1 when $x_i(t) = 1$ in $I_{i1}(j)$. We denote the total time duration when $\Delta_{i1}(t)$ is equal to 1 as $T_{e,1}(i,j)$ during the jth cycle where $T_{e,1}(i,j) = \sum_{\ell=1}^{K_1} T_m(i,\ell)$. Thus, we have $\mathbb{E}[T_{e,1}(i)] = \frac{1}{s_i}$. Then, using ergodicity, similar to [101], Δ_{i1} is equal to

$$\Delta_{i1} = \frac{\mathbb{E}[T_{e,1}(i)]}{\mathbb{E}[I_i]} = \frac{\mathbb{E}[T_{e,1}(i)]}{\mathbb{E}[I_{i1}] + \mathbb{E}[I_{i2}]}. \tag{10.13}$$

Thus, we have

$$\Delta_{i1} = \frac{\mu_i\lambda_i}{\mu_i + \lambda_i}\frac{c_i}{\mu_i c_i + \lambda_i s_i + c_i s_i}. \tag{10.14}$$

Next, we find Δ_{i2}. We note that $\Delta_{i2}(t)$ is equal to 1 when $x_i(t) = 0$ in $I_{i2}(j)$ and is always equal to 0 during $I_{i1}(j)$. Similarly, we denote the total time duration where $\Delta_{i2}(t)$ is equal to 1 in the jth cycle $I_i(j)$ as $T_{e,2}(i,j)$ which is equal to $T_{e,2}(i,j) = \sum_{\ell=1}^{K_2} T_u(i,\ell)$. Thus, we have $\mathbb{E}[T_{e,2}(i)] = \frac{1}{c_i}$. Then, similar to Δ_{i1} in (10.13), Δ_{i2} is equal to

$$\Delta_{i2} = \frac{\mu_i \lambda_i}{\mu_i + \lambda_i} \frac{s_i}{\mu_i c_i + \lambda_i s_i + c_i s_i}. \tag{10.15}$$

By using (10.5), (10.14), and (10.15), we obtain Δ_i as

$$\Delta_i = \frac{\mu_i \lambda_i}{\mu_i + \lambda_i} \frac{\theta c_i + (1-\theta) s_i}{\mu_i c_i + \lambda_i s_i + c_i s_i}. \tag{10.16}$$

Then, by inserting (10.16) in (10.7), we obtain Δ. In the next section, we solve the optimization problem in (10.8).

10.4 Optimization of Average Difference

In this section, we solve the optimization problem in (10.8). Using Δ_i in (10.16) in (10.7), we rewrite (10.8) as

$$\begin{aligned} \min_{\{s_i, c_i\}} \quad & \sum_{i=1}^{n} \frac{\mu_i \lambda_i}{\mu_i + \lambda_i} \frac{\theta c_i + (1-\theta) s_i}{\mu_i c_i + \lambda_i s_i + c_i s_i} \\ \text{s.t.} \quad & \sum_{i=1}^{n} s_i + \sum_{i=1}^{n} c_i \leq C \\ & s_i \geq 0, \quad c_i \geq 0, \quad i = 1, \dots, n, \end{aligned} \tag{10.17}$$

We define the Lagrangian function [149] for (10.17) as

$$\mathcal{L} = \sum_{i=1}^{n} \frac{\mu_i\lambda_i}{\mu_i+\lambda_i} \frac{\theta c_i + (1-\theta)s_i}{\mu_i c_i + \lambda_i s_i + c_i s_i} + \beta\left(\sum_{i=1}^{n} s_i + c_i - C\right) - \sum_{i=1}^{n} \nu_i s_i - \sum_{i=1}^{n} \eta_i c_i, \tag{10.18}$$

where $\beta \geq 0$, $\nu_i \geq 0$, and $\eta_i \geq 0$. The KKT conditions are

$$\frac{\partial \mathcal{L}}{\partial s_i} = \frac{\mu_i\lambda_i c_i}{\mu_i+\lambda_i} \frac{(1-\theta)\mu_i - \theta(c_i+\lambda_i)}{(\mu_i c_i + \lambda_i s_i + s_i c_i)^2} + \beta - \nu_i = 0, \tag{10.19}$$

$$\frac{\partial \mathcal{L}}{\partial c_i} = \frac{\mu_i\lambda_i s_i}{\mu_i+\lambda_i} \frac{\theta\lambda_i - (1-\theta)(\mu_i+s_i)}{(\mu_i c_i + \lambda_i s_i + s_i c_i)^2} + \beta - \eta_i = 0, \tag{10.20}$$

for all i. The complementary slackness conditions are

$$\beta\left(\sum_{i=1}^{n} s_i + c_i - C\right) = 0, \quad \nu_i s_i = 0, \quad \eta_i c_i = 0. \tag{10.21}$$

First, we find s_i. From (10.19), we have

$$(\mu_i c_i + \lambda_i s_i + s_i c_i)^2 = \frac{\mu_i\lambda_i c_i}{\mu_i+\lambda_i} \frac{\theta(c_i+\lambda_i) - (1-\theta)\mu_i}{\beta - \nu_i}. \tag{10.22}$$

When $\theta(c_i+\lambda_i) \geq (1-\theta)\mu_i$, we solve (10.22) for s_i as

$$s_i = \frac{\mu_i c_i}{\lambda_i + c_i}\left(\sqrt{\frac{1}{\mu_i c_i}\frac{\lambda_i}{\mu_i+\lambda_i}\frac{\theta(c_i+\lambda_i)-(1-\theta)\mu_i}{\beta}} - 1\right)^+, \tag{10.23}$$

where we used the fact that we either have $s_i > 0$ and $\nu_i = 0$, or $s_i = 0$ and $\nu_i \geq 0$, due to (10.21). Here, $(\cdot)^+ = \max(\cdot, 0)$.

Finally, when $\theta(c_i + \lambda_i) < (1-\theta)\mu_i$, we have $\frac{\partial \Delta_i}{\partial s_i} > 0$, and thus it is optimal to choose $s_i = 0$ as our aim is to minimize Δ in (10.7). In this case, when $s_i = 0$, we have $\Delta_i = \frac{\theta\lambda_i}{\mu_i+\lambda_i}$ which is independent of the value of c_i. As we obtain the same Δ_i for all values of c_i, and the total update rate is limited, i.e., $\sum_{i=1}^n s_i + c_i \leq C$, in this case, it is optimal to choose $c_i = 0$ as well (i.e., when $s_i = 0$).

Next, we find c_i. From (10.20), we have

$$(\mu_i c_i + \lambda_i s_i + s_i c_i)^2 = \frac{\mu_i \lambda_i s_i}{\mu_i + \lambda_i} \frac{(1-\theta)(\mu_i + s_i) - \theta\lambda_i}{\beta - \eta_i}. \tag{10.24}$$

When $(1-\theta)(\mu_i + s_i) \geq \theta\lambda_i$, we solve (10.24) for c_i as

$$c_i = \frac{\lambda_i s_i}{\mu_i + s_i}\left(\sqrt{\frac{1}{\lambda_i s_i}\frac{\mu_i}{\mu_i+\lambda_i}\frac{(1-\theta)(s_i+\mu_i) - \theta\lambda_i}{\beta}} - 1\right)^+, \tag{10.25}$$

where we used the fact that we either have $c_i > 0$ and $\eta_i = 0$, or $c_i = 0$ and $\eta_i \geq 0$, due to (10.21).

Similarly, when $(1-\theta)(s_i + \mu_i) < \theta\lambda_i$, we have $\frac{\partial \Delta_i}{\partial c_i} > 0$. Thus, in this case, it is optimal to choose $c_i = 0$. When $c_i = 0$, we have $\Delta_i = \frac{(1-\theta)\mu_i}{\mu_i+\lambda_i}$ which is independent of the value of s_i. Thus, it is optimal to choose $s_i = 0$ when $c_i = 0$.

From (10.23), if $\frac{1}{\mu_i c_i}\frac{\lambda_i}{\mu_i+\lambda_i}(\theta(c_i+\lambda_i) - (1-\theta)\mu_i) \leq \beta$, we must have $s_i = 0$. Thus, for a given c_i, the optimal test rate allocation policy for s_i is a threshold policy where s_i's with small $\frac{1}{\mu_i c_i}\frac{\lambda_i}{\mu_i+\lambda_i}(\theta(c_i+\lambda_i) - (1-\theta)\mu_i)$ are equal to zero. Similarly, from (10.25), if $\frac{1}{\lambda_i s_i}\frac{\mu_i}{\mu_i+\lambda_i}((1-\theta)(s_i+\mu_i) - \theta\lambda_i) \leq \beta$, we must have $c_i = 0$. Thus, for a given s_i, the optimal policy to determine c_i is a threshold policy where c_i's with small $\frac{1}{\lambda_i s_i}\frac{\mu_i}{\mu_i+\lambda_i}\left((1-\theta)(s_i+\mu_i)-\theta\lambda_i\right)$ are equal to zero.

Next, we show that in the optimal policy, if $s_i>0$ and $c_i>0$ for some i, then the total test rate constraint must be satisfied with equality, i.e., $\sum_{i=1}^n s_i+c_i=C$.

Lemma 10.1 *In the optimal policy, if $s_i>0$ and $c_i>0$ for some i, then we have $\sum_{i=1}^n s_i+c_i=C$.*

Proof: The derivatives of Δ_i with respect to s_i and c_i are

$$\frac{\partial \Delta_i}{\partial s_i}=\frac{\mu_i\lambda_i c_i}{\mu_i+\lambda_i}\frac{(1-\theta)\mu_i-\theta(c_i+\lambda_i)}{(c_i\mu_i+s_ic_i+\lambda_is_i)^2}, \tag{10.26}$$

$$\frac{\partial \Delta_i}{\partial c_i}=\frac{\mu_i\lambda_i s_i}{\mu_i+\lambda_i}\frac{\theta\lambda_i-(1-\theta)(s_i+\mu_i)}{(c_i\mu_i+s_ic_i+\lambda_is_i)^2}. \tag{10.27}$$

We note that $s_i>0$ in (10.23) implies that $\theta(c_i+\lambda_i)>(1-\theta)\mu_i$. In this case, we have $\frac{\partial\Delta_i}{\partial s_i}<0$. Similarly, $c_i>0$ in (10.25) implies that $(1-\theta)(s_i+\mu_i)>\theta\lambda_i$. Thus, we have $\frac{\partial\Delta_i}{\partial c_i}<0$. Therefore, in the optimal policy, if we have $s_i>0$ and $c_i>0$ for some i, then we must have $\sum_{i=1}^n s_i+c_i=C$. Otherwise, we can further decrease Δ in (10.7) by increasing c_i or s_i. ■

Next, we propose an alternating minimization based algorithm for finding s_i and c_i. For this purpose, for given initial (s_i,c_i) pairs, we define ϕ_i as

$$\phi_i=\begin{cases}\frac{1}{\mu_ic_i}\frac{\lambda_i}{\mu_i+\lambda_i}(\theta(c_i+\lambda_i)-(1-\theta)\mu_i),\ i=1,\dots,n,\\[2ex] \frac{1}{\lambda_is_i}\frac{\mu_i}{\mu_i+\lambda_i}((1-\theta)(s_i+\mu_i)-\theta\lambda_i),\ i=n+1,\dots,2n.\end{cases} \tag{10.28}$$

Then, we define u_i as

$$u_i = \begin{cases} \frac{\mu_i c_i}{\lambda_i + c_i}\left(\sqrt{\frac{\phi_i}{\beta}} - 1\right)^+, & i = 1, \ldots, n, \\ \frac{\lambda_i s_i}{\mu_i + s_i}\left(\sqrt{\frac{\phi_i}{\beta}} - 1\right)^+, & i = n+1, \ldots, 2n. \end{cases} \tag{10.29}$$

From (10.23) and (10.25), $s_i = u_i$ and $c_i = u_{n+i}$, for $i = 1, \ldots, n$.

Next, we find s_i and c_i by determining β in (10.29). First, assume that, in the optimal policy, there is an i such that $s_i > 0$ and $c_i > 0$. Thus, by Lemma 10.1, we must have $\sum_{i=1}^n s_i + c_i = C$. We initially take random (s_i, c_i) pairs such that $\sum_{i=1}^n s_i + c_i = C$. Then, given the initial (s_i, c_i) pairs, we immediately choose $u_i = 0$ for $\phi_i < 0$. For the remaining u_i with $\phi_i \geq 0$, we apply a solution method similar to that in [101]. By assuming $\phi_i \geq \beta$, i.e., by disregarding $(\cdot)^+$ in (10.29), we solve $\sum_{i=1}^{2n} u_i = C$ for β. Then, we compare the smallest ϕ_i which is larger than zero in (10.28) with β. If we have $\phi_i \geq \beta$, then it implies that $u_i \geq 0$ for all remaining i. Thus, we have obtained u_i values for given initial (s_i, c_i) pairs. If the smallest ϕ_i which is larger than zero is smaller than β, then the corresponding u_i is negative and we should choose $u_i = 0$ for the smallest non-negative ϕ_i. Then, we repeat this procedure until the smallest non-negative ϕ_i is larger than β. After determining all u_i, we obtain $s_i = u_i$ and $c_i = u_{n+i}$ for $i = 1, \ldots, n$. Then, with the updated values of (s_i, c_i) pairs, we keep finding u_i's until the KKT conditions in (10.19) and (10.20) are satisfied.

We note that for indices (persons) i for which (s_i, c_i) are zero, the health care provider does not perform any tests, and maps these people as either always infected, i.e., $\hat{x}_i(t) = 1$ for all t, or always healthy, i.e., $\hat{x}_i(t) = 0$. If $\hat{x}_i(t) = 0$ for all t, $\Delta_i = \frac{\theta\lambda_i}{\mu_i+\lambda_i}$, and if $\hat{x}_i(t) = 1$ for all t, $\Delta_i = \frac{(1-\theta)\mu_i}{\mu_i+\lambda_i}$. Thus, for such i, the health care provider should choose $\hat{x}_i(t) = 0$ for all t, if $\frac{\theta\lambda_i}{\mu_i+\lambda_i} < \frac{(1-\theta)\mu_i}{\mu_i+\lambda_i}$, and should choose $\hat{x}_i(t) = 1$ for all t, otherwise, without performing any tests.

Finally, we note that the problem in (10.17) is not a convex optimization problem as the objective function is not jointly convex in s_i and c_i. Therefore, the solutions obtained via the proposed method may not be globally optimal. For that reason, we choose different initial starting points and apply the proposed alternating minimization based algorithm and choose the solution that achieves the smallest Δ in (10.7).

10.5 Numerical Results

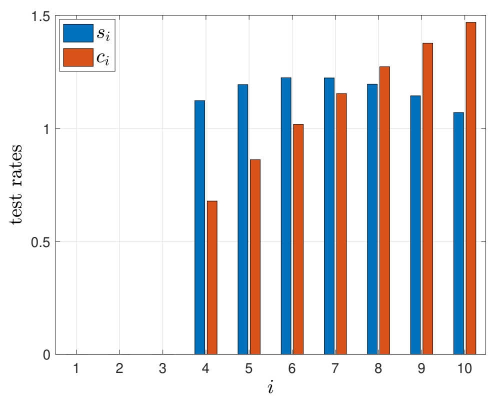
In this section, we provide four numerical results. For these examples, we take λ_i as

$$\lambda_i = ar^i, \quad i = 1, \ldots, n, \tag{10.30}$$

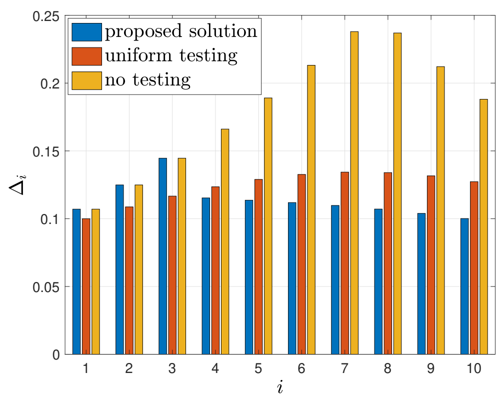
where $r = 0.9$ and a is such that $\sum_{i=1}^n \lambda_i = 6$. Also, we take μ_i as

$$\mu_i = bq^i, \quad i = 1, \ldots, n, \tag{10.31}$$

where $q = 1.1$ and b is such that $\sum_{i=1}^n \mu_i = 4$. Since λ_i in (10.30) decreases with i, people with lower indices get infected more quickly compared to people with higher indices. Since μ_i in (10.31) increases with i, people with higher indices recover more



(a)



(b)

Figure 10.4: (a) Test rates s_i and c_i, (b) corresponding average difference Δ_i.

quickly compared to people with lower indices. Thus, low index people get infected quickly and get well slowly.

In the first example, we take the total number of people as $n = 10$, the total test rate as $C = 16$, and $\theta = 0.5$. We start with randomly chosen s_i and c_i such that $\sum_{i=1}^{n} s_i + c_i = 16$, and apply the alternating minimization based method proposed in Section 10.4. We repeat this process for 30 different initial (s_i, c_i) pairs and choose the solution that gives the smallest Δ. In Fig. 10.4(a), we observe that the first three people are never tested by the health care provider. We note that s_i, which is the test rate when $\hat{x}_i(t) = 0$, initially increases with i but then decreases with i. This means that people who get infected rarely are tested less frequently when they are marked as healthy. Similarly, we observe in Fig. 10.4(a) that c_i, which is the test rate when $\hat{x}_i(t) = 1$, monotonically increases with i. In other words, people who recover from the virus quickly are tested more frequently when they are marked infected.

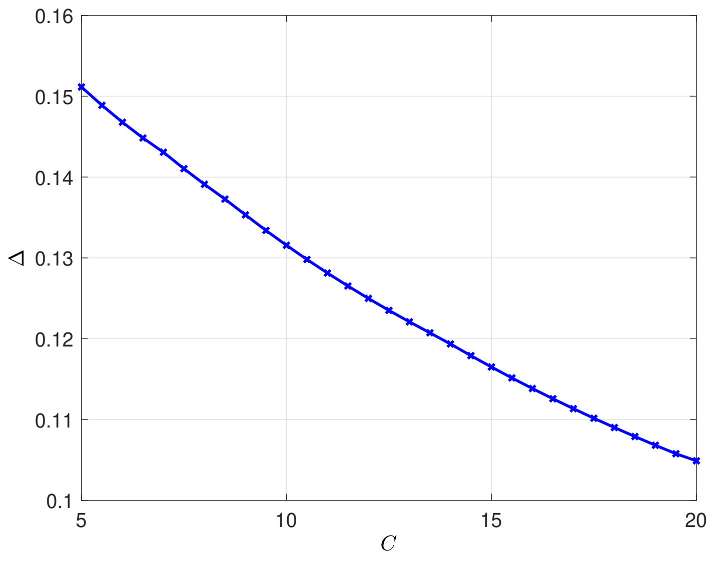


Figure 10.5: The average difference Δ with respect to total test rate C.

In Fig. 10.4(b), we plot Δ_i resulting from the solution found from the proposed algorithm, Δ_i when the health care provider applies tests to everyone in the population uniformly, i.e., $s_i = c_i = \frac{C}{2n}$ for all i, and Δ_i when the health care provider applies no tests, i.e., $s_i = c_i = 0$ for all i. In the case of no tests, we have $\Delta_i = \min\{\frac{\theta\lambda_i}{\mu_i+\lambda_i}, \frac{(1-\theta)\mu_i}{\mu_i+\lambda_i}\}$. We observe in Fig. 10.4(b) that the health care provider applies tests on people whose Δ_i can be reduced the most as opposed to uniform testing where everyone is tested equally. Thus, the first three people who have the smallest Δ_i are not tested by the health care provider. With the proposed solution, by not testing the first three people, Δ_i are further reduced for the remaining people compared to uniform testing. For the people who are not tested, the health care provider chooses $\hat{x}_i(t) = 1$ all the time, i.e., marks these people always sick as $\frac{\theta\lambda_i}{\mu_i+\lambda_i} > \frac{(1-\theta)\mu_i}{\mu_i+\lambda_i}$. This is expected as these people have high λ_i and low μ_i, i.e., they are infected easily and they stay sick for a long time.

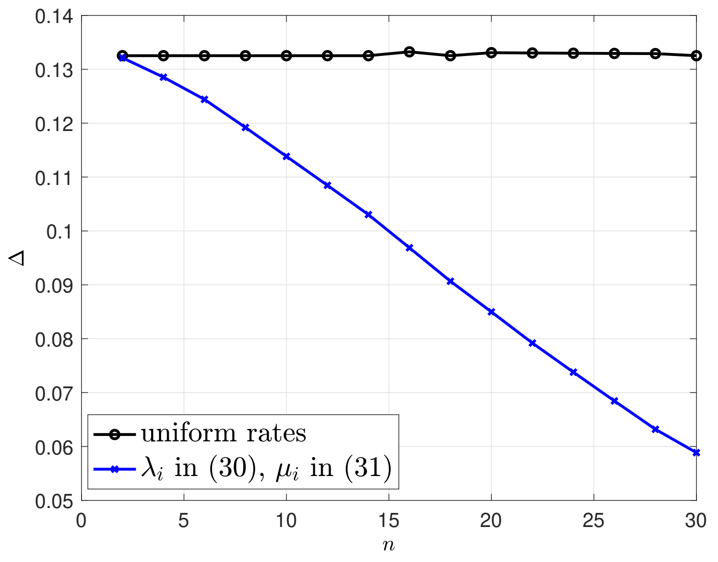
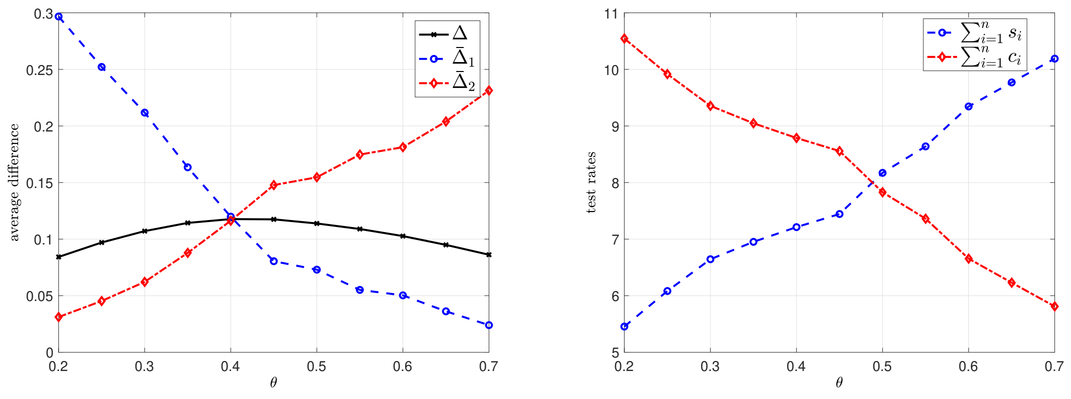


Figure 10.6: The average difference Δ with respect to number of people n. We use uniform infection and healing rates, i.e., $\lambda_i = \frac{6}{n}$ and $\mu_i = \frac{4}{n}$ for all i, and also λ_i in (10.30) and μ_i in (10.31) with $\sum_{i=1}^{n} \lambda_i = 6$ and $\sum_{i=1}^{n} \mu_i = 4$.

In the second example, we use the same set of variables except for the total test rate C. We vary the total test rate C in between 5 and 20. We plot Δ with respect to C in Fig. 10.5. We observe that Δ decreases with C. Thus, with higher total test rates, the health care provider can tract the infection status of the population better as expected.

In the third example, we use the same set of variables except for the total number of people n. In addition, we also use uniform infection and healing rates, i.e., $\lambda_i = \frac{6}{n}$ and $\mu_i = \frac{4}{n}$ for all i, for comparison with λ_i in (10.30) and μ_i in (10.31), while keeping the total infection and healing rates the same, i.e., $\sum_{i=1}^{n} \lambda_i = 6$ and $\sum_{i=1}^{n} \mu_i = 4$, for both cases. We vary the number of people n from 2 to 30. We observe in Fig. 10.6 that when the infection and healing rates are uniform in the population, the health care provider can track the infection status with the same



(a) (b)

Figure 10.7: (a) Δ in (10.7), $\bar{\Delta}_1$ which is $\frac{1}{n}\sum_{i=1}^{n}\Delta_{i1}$, and $\bar{\Delta}_2$ which is $\frac{1}{n}\sum_{i=1}^{n}\Delta_{i2}$, (b) corresponding total test rates $\sum_{i=1}^{n} s_i$ and $\sum_{i=1}^{n} c_i$.

efficiency, even though the population size increases (while keeping the total infection and healing rates fixed). For the case of λ_i in (10.30) and μ_i in (10.31), when we increase the population size, we increase the number of people who rarely get sick, i.e., people with high i indices, and also people who rarely heal from the disease, i.e., people with small i indices. Thus, it gets easier for the health care provider to track the infection status of the people. That is why when we use λ_i in (10.30) and μ_i in (10.31), we observe in Fig. 10.6 that the health care provider can track the infection status of the people better, even though the population size increases.

In the fourth example, we use the same set of variables as the first example except for the importance factor θ. Here, we vary θ in between 0.2 to 0.7. We plot Δ in (10.7), $\bar{\Delta}_1$ which is $\bar{\Delta}_1 = \frac{1}{n}\sum_{i=1}^{n}\Delta_{i1}$, and $\bar{\Delta}_2$ which is $\bar{\Delta}_2 = \frac{1}{n}\sum_{i=1}^{n}\Delta_{i2}$ in Fig. 10.7(a). Note that $\bar{\Delta}_1$ represents the average difference when people are infected, but they have not been detected by the health care provider, and $\bar{\Delta}_2$ represents the average difference when people have recovered, but the health care provider still marks them as infected. Note that when θ is high, we give importance to minimization of $\bar{\Delta}_1$, i.e., the early detection of people with infection, and when θ is low, we give importance to minimization of $\bar{\Delta}_2$, i.e., the early detection of people who recovered from the disease. That is why we observe in Fig. 10.7(a) that $\bar{\Delta}_1$ decreases with θ while $\bar{\Delta}_2$ increases with θ.

We plot the total test rates $\sum_{i=1}^{n} s_i$ and $\sum_{i=1}^{n} c_i$ in Fig. 10.7(b). We observe in Fig. 10.7(b) that if it is more important to detect the infected people, i.e., if θ is high, then the health care provider should apply higher test rates to people who are marked as healthy. In other words, $\sum_{i=1}^{n} s_i$ increases with θ. Similarly, if it is more important to detect people who recovered from the disease, then the health care provider should apply high test rates to people who are marked as infected. That is, $\sum_{i=1}^{n} c_i$ is high when θ is low. Therefore, depending on the priorities of the health care provider, a suitable θ needs to be chosen.

10.6 Conclusion

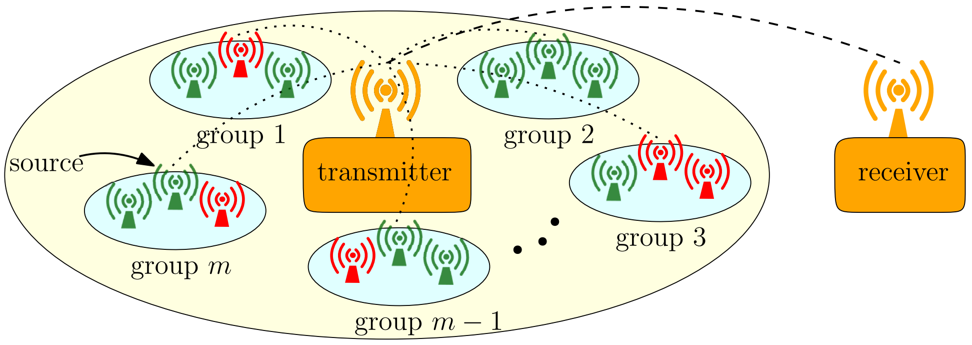
In this chapter, we considered timely tracking of infection status of individuals in a population. For exponential infection and healing processes with given rates, we determined the rates of exponential testing processes. We observed in numerical results that the test rates depend on individuals' infection and healing rates, the individuals' last known state of healthy or infected, as well as the health care provider's priorities of detecting infected people or recovered people more quickly.

CHAPTER 11

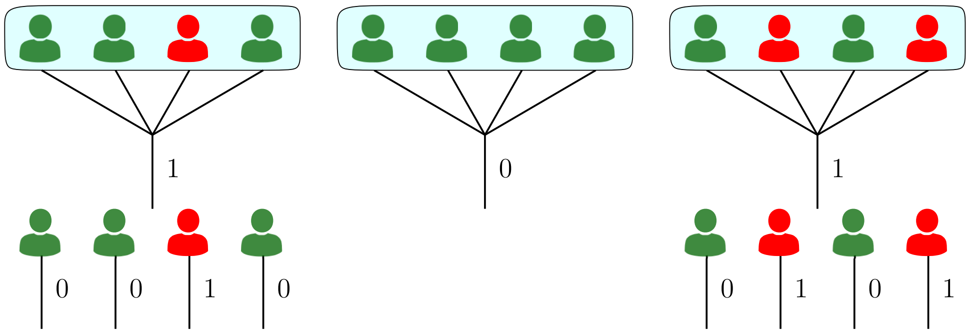
Age-Efficient Scheduling for Binary-Valued Information Sources: Group Updating

11.1 Introduction

In this chapter, we consider two different problems with similar system models: anomaly detection in sensor networks and testing for infections in human populations. In the anomaly detection problem, n sensor nodes monitor a region and make measurements for an anomaly (e.g., fire, chemical spills, etc.) and report their measurements to a central location; see Fig. 11.1(a). Each sensor node detects an anomaly with probability p independent of others. In the infection testing problem, there are n individuals each of whom is infected with probability p independent of others, and their infection status needs to be tallied at a central location; see Fig. 11.1(b). In both problems, we want to identify the anomaly/infection status of each node as timely as possible in order to take necessary actions as quickly as possible, e.g., control the fire or isolate/treat the infected persons. For a measure of timeliness, we use age of information, which keeps track of the time elapsed since the last time the status of a node is updated.



(a) An anomaly detection system with multiple sensor nodes. Sensors in red indicate an anomaly and sensors in green indicate no anomaly.



(b) An infection detection system in a human population. Persons in red are infected and persons in green are not infected.

Figure 11.1: System models considered in this chapter.

Inspired by the *group testing* approach introduced in [159], we develop a *group updating* approach to maintain timely status updates at the central location. To that end, we divide n nodes into groups of k nodes each. In the case of anomaly detection, a local transmitter collects anomaly status of all nodes within the group. If there is no anomaly detected within the group, the local transmitter sends a single 0 to the central location. The central location, then, knows the status of all nodes within the group. On the other hand, if there is at least one anomaly detected within the group, the local transmitter sends a 1 to the central location. The central location, then, knows that there is at least one anomalous reading within the group.

The local transmitter then sends the individual measurements of the sensors (0s and 1s) to the central location one-by-one. Similarly, in the case of testing humans for infection, we divide n individuals into groups of k each. Within each group, we mix the test samples of the individuals and perform a single test. If the test result is a 0, we know that no one within the group is infected. If the test result is a 1, then, we know that at least one person within the group is infected. In the latter case, we test each person within the group individually one-by-one.

In the proposed group updating method, the group size k plays an important role in the performance of the system, i.e., in the resulting age. If k is too large, then the first update will likely result in a 1, and we will need to proceed to update the status of each node within the group one-by-one. This will increase the update duration, and hence, the age. On the other hand, if k is too small, then this will result in too many groups, and therefore, too many updates within an update cycle. This will increase the age as well. Thus, there is an optimum group size k, which is not too small, not too large. Here, we find that optimum size for given n and p.

With this work, we are bringing age of information as a measure of timeliness to anomaly detection and testing for infections. Specifically relevant to our case is the setting of multi-source systems, where maximum age first (MAF) [160], maximum age difference (MAD) [161], Whittle index [3,59,162], slotted ALOHA with threshold [163], hierarchical cooperation [65] have been used to achieve good age performance. Different from most of these works, where only one source can be updated at a time, with the proposed group updating approach, we allow all sources in a group to be updated simultaneously with a single status update.

In this chapter, we introduce a group updating approach where if all updates from the sources in the same group are 0, then the transmitter sends only a single status update representing the entire group, otherwise, the transmitter sends an update indicating that there is at least one 1 in the group, and proceeds to send all individual updates within the group one-by-one. For this updating method, for arbitrary n and p, we first find an analytical expression for the average age, which depends on the group size k. For given n and p, we find the optimal group size k that minimizes the age. Next, we compare the performance of the proposed group updating policy with the performances of the traditional scheduling methods, and observe that the proposed group updating policy achieves a lower age than the existing schemes when p is small. In addition, we compare the optimal group size k in the group updating problem and in the group testing problem in [159] and observe that they are different in general indicating the difference of the metrics used.

11.2 System Model

We consider a system with n sources/nodes. We divide the n sources into groups of size k, where $m = \frac{n}{k}$ is the number of groups. Without loss of generality, we assume that k divides n, and thus, m is an integer. We denote the status of the jth source in the ith group in the ℓth update cycle by $X_{ij}(\ell)$, where $i = 1, \dots, m$, $j = 1, \dots, k$, and $\ell \geq 1$. $X_{ij}(\ell)$ is an independent and identically distributed (i.i.d.)

binary random variable for all i, j and ℓ, with distribution,

$$X_{ij}(\ell) = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1-p, \end{cases} \tag{11.1}$$

where a status 1 indicates an anomaly/infection, and a status 0 indicates no anomaly /infection.

Let $S_{ij}(\ell)$ denote the service time for the status update of the jth source in the ith group in the ℓth update cycle. This is the time it takes for the status of the node to go through the system and be tallied at the central location. Note that if the status of all nodes in the ith group is 0, then the service time for all nodes in this group is equal to 1, as in this case, for the anomaly detection problem, the local transmitter needs to send a single 0 to convey the status of all nodes, and in the infection testing problem, a single test will determine the infection status of all nodes in the group. On the other hand, if any one of the sources in the ith group generates 1 as a status update, the service time for the jth source in the ith group will be equal to $j+1$, as in this case, an initial status update is sent representing the entire group, $j-1$ status updates are sent for the sources before source j, and a final update is sent for source j itself. Thus, the service time for the jth node in group i is a random variable with distribution,

$$S_{ij}(\ell) = \begin{cases} 1, & \text{with probability } (1-p)^k, \\ j+1, & \text{with probability } 1-(1-p)^k. \end{cases} \tag{11.2}$$

The service time of the entire ith group in the ℓth update cycle, denoted by $W_i(\ell)$, is equal to the service time of the last source in the ith group,

$$W_i(\ell) = S_{ik}(\ell), \quad i = 1, \dots, m. \tag{11.3}$$

As the central location wants to get timely updates from all sources, we track the age of each source at the central location separately. We denote the instantaneous age of source j in group i at time t by $a_{ij}(t)$, with $a_{ij}(0) = 0$. Age of each source at the central location increases linearly in time and drops to the age of the most recently received update once an update is received. The long term average age of node j in group i is given by,

$$\Delta_{ij} = \lim_{T \to \infty} \frac{1}{T} \int_0^T a_{ij}(t) dt. \tag{11.4}$$

The overall average age of all sources Δ is equal to,

$$\Delta = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{k} \Delta_{ij}. \tag{11.5}$$

Our aim is to find the optimal group size k^* that minimizes the average age of all sources Δ, i.e.,

$$k^* = \arg\min_{\{k\}} \Delta. \tag{11.6}$$

In Section 11.3, we first find the average age, Δ, in (11.5).

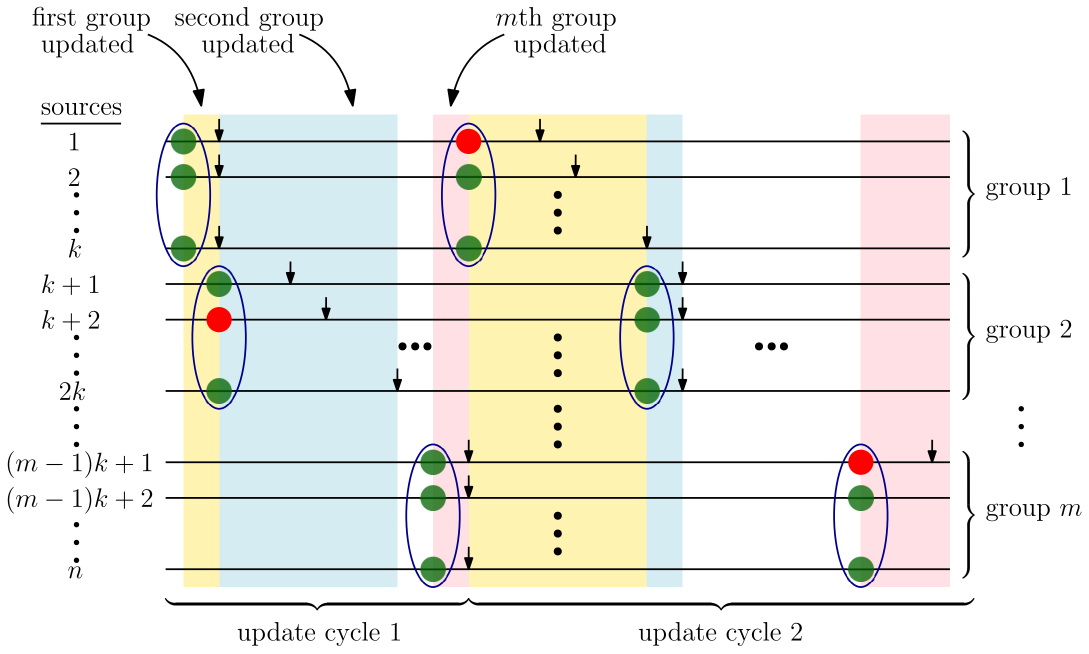


Figure 11.2: A sample update generation and update delivery timeline. Lines 1 through n denote the nodes. Lines 1 through k denote the nodes in group 1. Green and red balls represent the anomaly/no anomaly status of each node. In update cycle 1, the yellow strip shows the time where the status of all nodes in group 1 is updated, the blue strip shows the time where the status of all nodes in group 2 is updated, and the pink strip shows the time where the status of all nodes in group m is updated. The process repeats itself in update cycle 2. Delivery times are marked by the downward arrows.

11.3 Average Age Analysis

With the group updating policy, the transmitter starts with sending updates from the sources in the first group. If all the updates from the first group are 0 (as shown with green balls in the first k lines in Fig. 11.2), then the transmitter sends a single 0 to update all the sources in the first group (that is why the delivery times of updates for all sources in the first group marked with arrows in Fig. 11.2 are equal to 1). After sending updates from the first group, the transmitter proceeds to send updates from the second group. If any one of the updates from the second group is equal to 1 (denoted by a red ball in the lines between lines $k+1$ and $2k$ in Fig. 11.2), then the transmitter first sends a 1 as a status update representing the entire group, and then sends individual updates from each source one-by-one. As shown in Fig. 11.2, the receiver gets the first update from the second group after 2 units of time. After sending updates from the second group, the transmitter proceeds to send updates from the third group, and so on, up until the mth (last) group. We call this entire time in which the status of all n sources are updated as update cycle 1 in Fig. 11.2. Once update cycle 1 ends, update cycle 2 starts all over again with all sources taking a new i.i.d. realization. In Fig. 11.2, in update cycle 1, the yellow vertical strip shows the time in which the status of all nodes in group 1 is updated, the blue strip shows the time in which the status of all nodes in group 2 is updated, so on so forth, and finally, the pink strip shows the time in which the status of all nodes in group m is updated.

Fig. 11.3 shows a sample age evolution curve for the jth source in the ith group at the central location, i.e., $a_{ij}(t)$. Here, $S_{ij}(\ell)$ defined in (11.2) denotes the service time of the jth source in the ith group in the ℓth update cycle. In addition, $W_{ij}(\ell)$ denotes the total waiting time until the ℓth update is generated after the service completion of the $(\ell-1)$th update for the same source. Thus, $W_{ij}(\ell)$ is given by,

$$W_{ij}(\ell) = \bar{W}_{ij}(\ell-1) + \sum_{r=i+1}^{m} W_r(\ell-1) + \sum_{r=1}^{i-1} W_r(\ell), \tag{11.7}$$

where $W_r(\ell)$ is given in (11.3), and $\bar{W}_{ij}(\ell-1)$ denotes the remaining service time of the ith group in the $(\ell-1)$th update cycle which is given by $\bar{W}_{ij}(\ell-1) =$

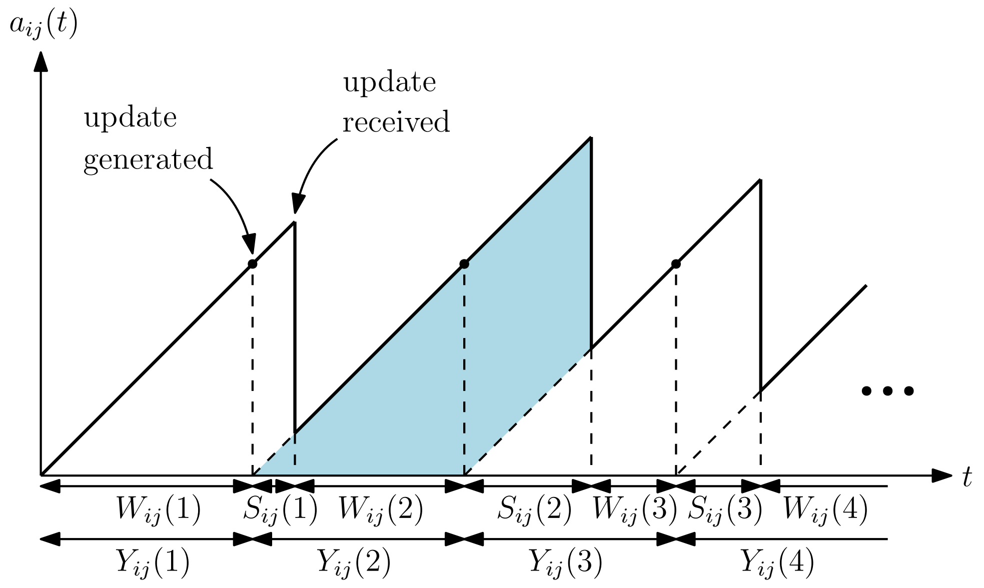


Figure 11.3: A sample age evolution $a_{ij}(t)$ at the central location.

$W_i(\ell-1) - S_{ij}(\ell-1)$.

We denote the length of the ℓth update cycle for the jth source in the ith group as $Y_{ij}(\ell) = S_{ij}(\ell-1) + W_{ij}(\ell)$ with $S_{ij}(0) = 0$ for convention. One can show that the long term average age Δ_{ij} given in (11.4) as in [10] is,

$$\Delta_{ij} = \lim_{N\to\infty} \frac{\frac{1}{N}\left(\frac{1}{2}\sum_{\ell=1}^{N+1} Y_{ij}(\ell)^2 + \sum_{\ell=1}^{N} Y_{ij}(\ell)S_{ij}(\ell)\right)}{\frac{1}{N}\sum_{\ell=1}^{N} Y_{ij}(\ell)}, \tag{11.8}$$

where N denotes the number of update cycles. We note that (11.8) can be written equivalently as,

$$\Delta_{ij} = \frac{\mathbb{E}[Y_{ij}^2]}{2\mathbb{E}[Y_{ij}]} + \mathbb{E}[S_{ij}]. \tag{11.9}$$

We note that the length of an update cycle Y_{ij} is equal to the service completion time of all the groups, i.e.,

$$Y_{ij} = S_{ij} + W_{ij} = \sum_{r=1}^{m} W_r. \tag{11.10}$$

Therefore, the variable Y_{ij} does not depend on i or j. We thus denote Y_{ij} with a single random variable Y, i.e., $Y = Y_{ij}$. On the other hand, from (11.2), S_{ij} depends on j, and we denote it by S_j. Then, the overall average age Δ in (11.5) is equal to,

$$\Delta = \frac{\mathbb{E}[Y^2]}{2\mathbb{E}[Y]} + \mathbb{E}[S], \tag{11.11}$$

where $\mathbb{E}[S] = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{k}\mathbb{E}[S_{ij}] = \frac{1}{k}\sum_{j=1}^{k}\mathbb{E}[S_j]$.

Now, using (11.2)-(11.3), we have $\mathbb{E}[W_r] = 1 + k\left(1-(1-p)^k\right)$ for all r. Thus, from (11.10),

$$\mathbb{E}[Y] = \frac{n}{k} + n\left(1-(1-p)^k\right). \tag{11.12}$$

In addition,

$$\mathbb{E}[Y^2] = n(n-k)(1-p)^{2k} + \frac{n^2(k+1)^2}{k^2} - n\left(2n\left(1+\frac{1}{k}\right) - k\right)(1-p)^k. \tag{11.13}$$

Further, from (11.2), we have,

$$\mathbb{E}[S_{ij}] = \mathbb{E}[S_j] = 1 + j(1-(1-p)^k), \tag{11.14}$$

and thus, we have,

$$\mathbb{E}[S] = 1 + \frac{k+1}{2}(1-(1-p)^k). \tag{11.15}$$

Hence, the overall average age Δ in (11.11) is

$$\begin{aligned}\Delta =& \frac{k^2(n-k)(1-p)^{2k} + n(k+1)^2}{2k + 2k^2\left(1-(1-p)^k\right)} - \frac{\left(2n\left(k+1\right)-k^2\right)\left(1-p\right)^k}{2+2k\left(1-(1-p)^k\right)} + 1 \\ &+ \frac{k+1}{2}(1-(1-p)^k). \end{aligned} \tag{11.16}$$

The overall average age in (11.16) depends on n, p and k. We find the optimal k that minimizes Δ numerically in Section 11.5.

11.4 Group Updating versus Group Testing

While the group testing and group updating policies are operationally similar, parameter selection, mainly selection of the group size in both problems, is different. In particular, in group testing, group size k is chosen to minimize the expected number of tests. In our terminology, expected number of tests corresponds to the expected length of an update cycle, i.e., $\mathbb{E}[Y]$, as the transmitter sends one status update at a time. Thus, group testing chooses the group size k^*_{gt} by solving,

$$k^*_{gt} = \arg \min_{\{k\in\mathbb{Z}^+\}} \mathbb{E}[Y], \tag{11.17}$$

where $\mathbb{E}[Y]$ is given in (11.12). In order for group testing to be more efficient than sequential updating of sources one-by-one, which uses n tests in an update cycle, we need $\mathbb{E}[Y] \leq n$, which implies $p \leq p_{gt}$, where

$$p_{gt} = 1 - \left(\frac{1}{k}\right)^{\frac{1}{k}}. \tag{11.18}$$

We note that p_{gt} attains its maximum value 0.3066 when $k = 3$. Thus, when $p > 0.3066$, group testing becomes inefficient compared to sequential updating of sources one-by-one.

Next, we find k^*_{gt} in (11.17) analytically. For that, we first relax the integer constraint on k. Then, by equating the derivative of $\mathbb{E}[Y]$ in (11.12) with respect to k to zero, we obtain,

$$\frac{\partial \mathbb{E}[Y]}{\partial k} = -\frac{n}{k^2} - n(1-p)^k \log(1-p) = 0, \tag{11.19}$$

which gives,

$$\frac{k}{2}\log(1-p)e^{\frac{k}{2}\log(1-p)} = -\frac{1}{2}\sqrt{-\log(1-p)}. \tag{11.20}$$

Note that (11.20) is in the form of $xe^x = y$, whose solutions for x are $x_1 = W_0(y)$ and $x_2 = W_{-1}(y)$ when $-\frac{1}{e} \leq y < 0$. Here, $W_0(\cdot)$ and $W_{-1}(\cdot)$ denote the principle and -1st branches of the Lambert W function, respectively [150]. Thus, when

$0 < p \leq 1 - e^{-\frac{4}{e^2}} = 0.418$, we have two solutions for (11.20) which are given by,

$$\alpha_1 = \frac{2}{\log(1-p)} W_0 \left(-\frac{1}{2}\sqrt{-\log(1-p)} \right), \tag{11.21}$$

$$\alpha_2 = \frac{2}{\log(1-p)} W_{-1} \left(-\frac{1}{2}\sqrt{-\log(1-p)} \right). \tag{11.22}$$

When $p > 0.418$, one can show that $\frac{\partial \mathbb{E}[Y]}{\partial k} < 0$, and thus, the optimal k is equal to n. However, as the group testing method becomes inefficient when $p > 0.3066$, we only need to consider the case when $0 < p \leq 0.418$, and thus, α_1 in (11.21) and α_2 in (11.22) always exist.

Thus, in order to find the optimal k, we need to check $k = \alpha_r^u$ where $\alpha_r^u = \min\{k | k \geq \alpha_r, k|n\}$ for $r = 1, 2$; $k = \alpha_r^\ell$ where $\alpha_r^\ell = \max\{k | k \leq \alpha_r, k|n\}$ for $r = 1, 2$; $k = 1$; and $k = n$. In other words, the optimal k is given by,

$$k_{gt}^* = \arg \min_{\{k \in \mathbb{K}\}} \mathbb{E}[Y], \tag{11.23}$$

where $\mathbb{K} = \{1, \alpha_1^\ell, \alpha_2^\ell, \alpha_1^u, \alpha_2^u, n\}$.

We perform a similar analysis for the group updating problem. Group updating chooses the group size k_{gu}^* by solving,

$$k_{gu}^* = \arg \min_{\{k \in \mathbb{Z}^+\}} \Delta, \tag{11.24}$$

where Δ is given in (11.16). In order for group updating to be more efficient than sequential updating, Δ in (11.16) needs to be smaller than $\Delta_{\text{round-robin}}$. For the round-robin (sequential) scheduling method, $\mathbb{E}[Y] = n$, $\mathbb{E}[Y^2] = n^2$, $\mathbb{E}[S] = 1$, and the overall average age from (11.11) is,

$$\Delta_{\text{round-robin}} = \frac{n}{2} + 1. \tag{11.25}$$

The condition $\Delta \leq \Delta_{\text{round-robin}}$ gives an upper bound for the probability p, which we denote by p_{gu}. In other words, when $p > p_{gu}$, group updating becomes inefficient compared to sequential updating. Further, by relaxing the integer constraint on k and equating the derivative of Δ in (11.16) with respect to k to 0, we can find the critical points where the age is minimized, and find k^*_{gu} analytically. Since Δ in (11.16) is an involved function of k, in this work, we do not pursue analytical results on p_{gu} and k^*_{gu}. Instead, we find k^*_{gu} for given of p and n, and examine p_{gu}, numerically, in the next section.

11.5 Numerical Results

In this section, we provide four numerical results to illustrate the performance of the proposed group updating method, and also to show its difference from the group testing method. In all the numerical results, we only consider k values that divide n. For example, if $n = 6$, we consider $k = 1, 2, 3, 6$.

In the first numerical example, we compare the performance of the proposed group updating method with the performances of the existing updating policies of MAF and MAD. Since after receiving each update, the age at the receiver goes down to 1, MAF and MAD scheduling policies become identical. In addition, as

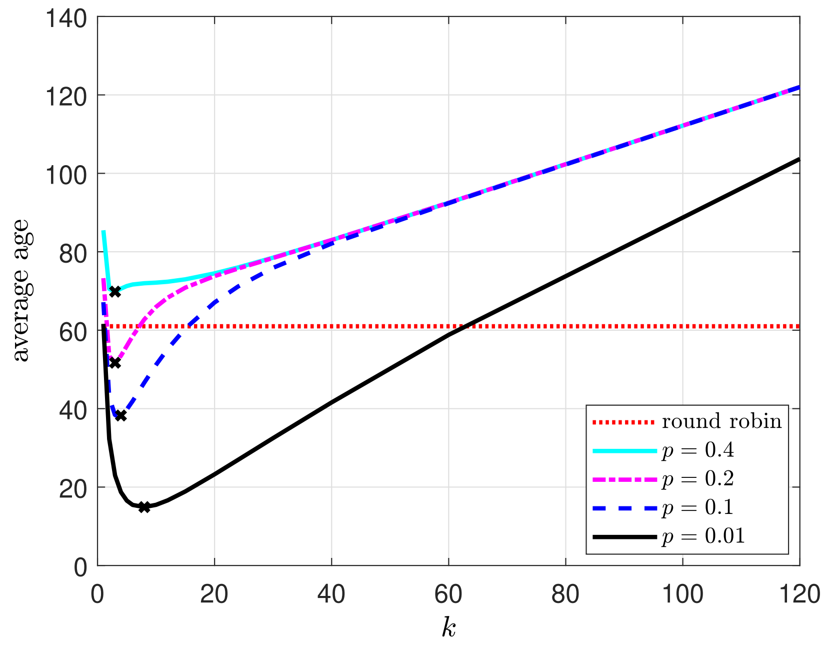


Figure 11.4: Average age versus group size with the proposed group updating method and the round robin method when $p = 0.01, 0.1, 0.2, 0.4$.

the ages of all sources start from zero, MAF and MAD policies become the same as the round-robin scheduling method. The average age for the round-robin scheme is given in (11.25). We note that $\Delta_{\text{round-robin}}$ increases linearly with n and does not depend on the probability p.

In the first numerical example, we take $n = 120$ and plot in Fig. 11.4 the average age Δ in (11.16) with respect to k when $p = 0.01, 0.1, 0.2, 0.4$, together with $\Delta_{\text{round-robin}}$ in (11.25). We observe in Fig. 11.4 that, for all values of p, the average age first decreases with k and then increases with k, as initially, increasing k decreases the number of groups, making group updating more efficient, but after a while, further increasing k decreases the likelihood of having all zero updates in a group, requiring many follow-up individual updates. Thus, there is a trade-off between these two opposing factors, and there is an optimum group size to minimize

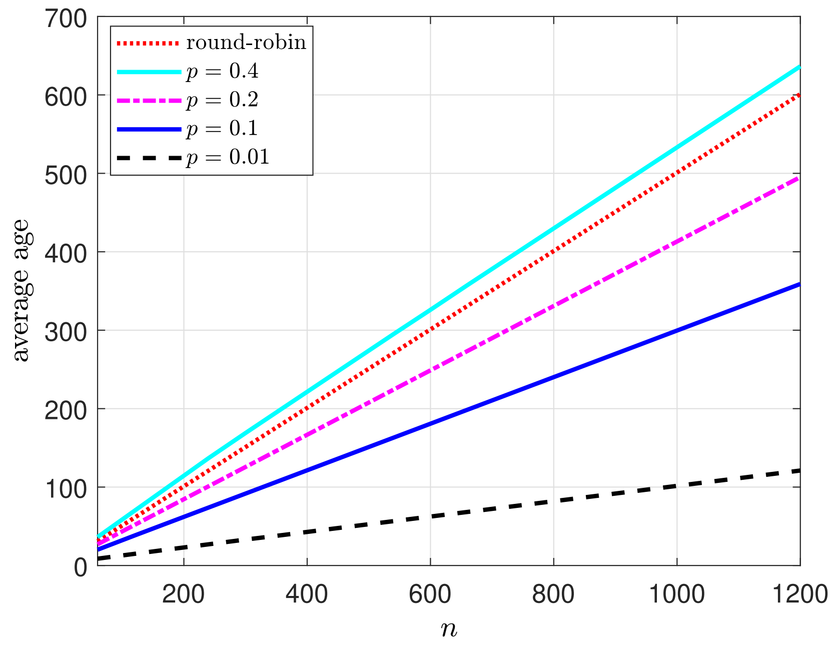


Figure 11.5: Average age versus population size with the proposed group updating method and the round robin method when $p = 0.01, 0.1, 0.2, 0.4$.

the average age. As marked with a cross in Fig. 11.4, when $p = 0.01$ the optimal group size is $k^*_{gu} = 8$; when $p = 0.1$ it is $k^*_{gu} = 4$; when $p = 0.2$ it is $k^*_{gu} = 3$; and when $p = 0.4$ it is $k^*_{gu} = 3$. We also observe that the group updating method becomes inefficient with increased p as it becomes more likely for the transmitter to send individual updates. When p is large enough, e.g., when $p = 0.4$, we observe in Fig. 11.4 that group updating becomes inefficient and does not improve the average age compared to the round-robin scheduling method.

In the second numerical example, we again take $p = 0.01, 0.1, 0.2, 0.4$, and plot in Fig. 11.5 the average age with respect to n, the population size, for n from 60 to 1200. For each value of p and n, we first find the optimal k^*_{gu} that achieves the minimum age, then plot that minimum age with respect to n. We observe in Fig. 11.5 that the average age increases linearly with the proposed group updating

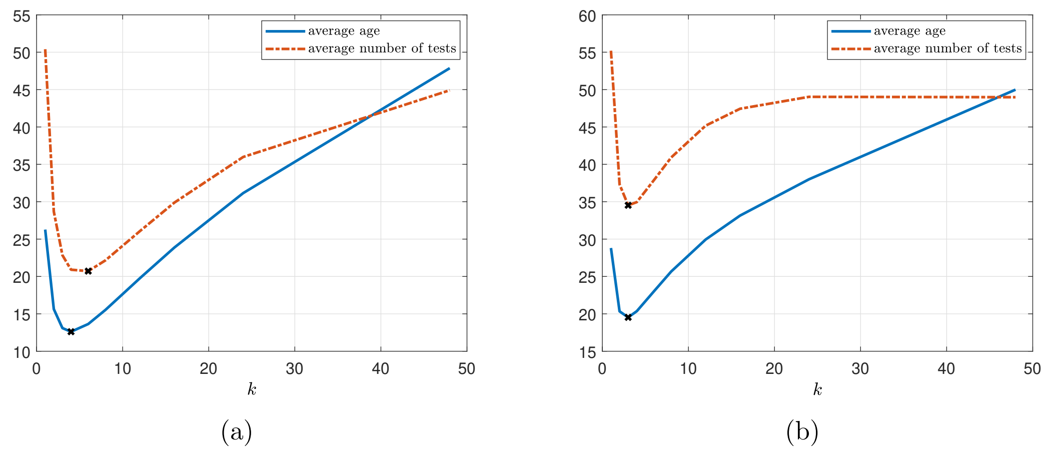


Figure 11.6: Average age for the group updating method and average number of updates for the group testing method with respect to k for $n = 48$ when (a) $p = 0.05$ and (b) $p = 0.15$.

method as with the round-robin scheduling method. Similar to the first numerical example, the average age increases with p as group updating becomes less efficient with larger p.

In the third numerical example, we examine the differences between the group updating problem and the group testing problem. For this numerical example, we take $n = 48$, $p = 0.05, 0.15$, and determine the optimal k values that minimize the average age and also the average number of updates. When p is small, e.g., when $p = 0.05$, we observe in Fig. 11.6(a) that the optimal group size that minimizes the average age is $k^*_{gu} = 4$, whereas the optimal group size that minimizes the average number of updates is $k^*_{gt} = 6$. This verifies that the group updating problem is different than the group testing problem. However, when p is relatively large, e.g., when $p = 0.15$, we observe in Fig. 11.6(b) that the optimal group sizes in both problems are equal $k^*_{gu} = k^*_{gt} = 3$. In other words, when p gets larger, the optimal k

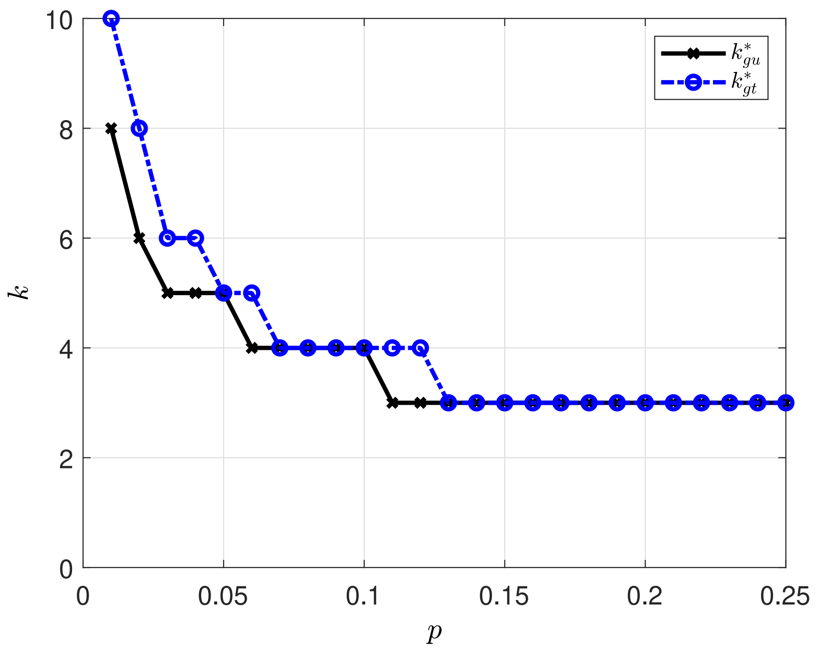


Figure 11.7: Optimum group sizes k^*_{gu} in the group updating problem and k^*_{gt} in the group testing problem for $n = 120$, for p from 0.01 to 0.25.

values for the group updating and group testing problems get closer to each other.

In the fourth numerical example, we examine k^*_{gu} and k^*_{gt} as a function of p. We take $n = 120$ and vary p between 0.01 and 0.25. We observe in Fig. 11.7 that both k^*_{gu} and k^*_{gt} decrease with probability p. With higher p, the sources in a group begin to generate more 1s as status updates, which results in sending more individual updates from the sources. Thus, decreasing the group size k^* in both of the problems helps counter the effects of increased p. Similar to the previous example, we observe in Fig. 11.7 that k^*_{gu} and k^*_{gt} are different when p is small, and become the same when $p \geq 0.13$ for this choice of n.

11.6 Conclusion

In this chapter, we considered the problem of timely group updating, where similar to group testing, the sources are divided into groups; if all updates within a group are negative, a single group update suffices; if at least one update is positive, this triggers a sequence of individual updates. For this updating scheme, we derived an analytical expression for the average age as a function of the group size k, the number of sources n, and the probability p. For given n and p, we found the optimal group size k that minimizes the age. We showed that when p is small, group updating performs better than sequential updating. We also showed that the optimal group sizes for group updating and group testing are different. This is because, while group testing aims to minimize the first moment of the length of an update cycle, group updating aims to minimize the age which depends on both the first and second moments of the length of an update cycle. An analogous observation was made in timely source coding versus traditional source coding, where the former depends on the first and second moments of the codeword length, while the latter depends only the first moment [80, 83].

CHAPTER 12

Conclusions

In this dissertation, we developed solutions for fundamental timely update delivery problems in communication networks with various system settings.

In Chapter 2, we introduced the concept of soft updates which begin reducing the age immediately but drop it gradually over time as opposed to traditional age metric where updates are countable (hard) and drop the age instantaneously (possibly after a delay). We studied two soft update regimes: exponentially and linearly decaying age models. For both of these models, we found the optimum start times for the soft updates and their corresponding optimum update durations to minimize the average age. We showed that the optimal policy is to utilize all updating opportunities by allocating equal amount of update duration for each update.

In Chapter 3, we modeled the distortion on updates as a decreasing function of the processing time that the transmitter spent to generate an update. We considered the problem of minimizing age of information at the receiver subject to a distortion constraint on each update. We considered constant and also age-dependent distortion constraints. For all these distortion constraints, we found the optimum time to request updates from the transmitter and their corresponding processing times to generate the updates. We showed that there is a trade-off between the age and distortion on the updates and for all these distortion constraints, the optimum processing times are equal to the minimum required processing duration that meets the distortion constraints.

In Chapter 4, we considered selective encoding policies that only encode the most probable status updates. For the remaining least probable updates, we studied the cases where these updates are never sent, probabilistically sent, represented by a designated empty symbol. For all these encoding schemes, we found the optimum number of encoded updates and as well as their corresponding age-optimal real valued codeword lengths.

In Chapter 5, we studied the problem of generating partial updates, which carry less information, but can be transmitted faster than the original updates. We minimized the age of information while maintaining a desired level of information fidelity between the original and the partial updates. We proposed an alternating minimization based method that produced a pmf for the partial updates and their corresponding age-optimum real-valued codeword lengths.

In Chapter 6, we considered the binary freshness metric for a caching system consisting of a source, cache(s), and a user. We characterized the binary freshness at the end user and proposed an alternating maximization based method to maximize the overall freshness at the end user subject to total update rate constraints on the cache(s), and the user.

In Chapter 7, different from our work in Chapter 6, we considered a caching system with a limited storage capacity for the cache. Here, the user can get the files from the cache at the expense of sometimes receiving an outdated version of the files; or get fresh files directly from the source at the expense of additional transmission times between the source and the user. For this problem, we found the optimum caching status of each file and their optimum update rates at the caches to maximize the overall freshness at the end user. We observed that rapidly changing files at the source and files with smaller transmission times can be directly obtained from the source when the total update rate and storage capacity of the cache are limited.

In Chapter 8, we studied binary freshness in structured gossip networks. Different from the works in Chapters 6 and 7, here, we allow the end nodes to share their local versions with each other via a process called gossiping. By using an SHS approach, we characterized the information freshness in arbitrarily connected gossip networks. When the number of nodes gets large, we showed that the binary freshness decreases to 0 as n^{-1} for both disconnected and ring networks, but with strictly higher freshness for the ring networks. When the update rates of the source and the nodes are sufficiently large, for fully connected networks, binary freshness decreases to 0 with a smaller rate. In addition, we considered clustered gossip networks and found the optimal cluster sizes numerically.

In Chapter 9, we considered the problem of timely updating of citation counts of a group of researchers under the limited total update rate constraint for Google Scholar. We modeled the citation arrival profile of each researcher as a counting process. We showed that the optimum update rates of researchers should be proportional to the square root of their citation arrival rates multiplied by their importance factors.

In Chapter 10, we considered the real-time timely tracking of infection status of people under limited total test rate constraints for the health care provider. For given infection and recovery rates of individuals, we found the rates of Poisson testing processes when the people are considered to be sick and healthy by the health care provider. When the total test rate is limited, we showed that some portion of the population may not be tested by the health care provider.

In Chapter 11, we considered a communication system with large number of sources that produce binary status updates that can indicate an anomaly or infection. Instead of sending updates sequentially, we proposed a group updating method inspired by the group testing approach, but with the aim of minimizing the average age over all sources. We showed that when the anomaly/infection probability is low, the proposed group updating method achieves lower average age than sequential updating methods.

The contents of Chapter 2 are published in [62, 63], Chapter 3 in [108, 109], Chapter 4 in [81–83], Chapter 5 in [87], Chapter 6 in [100, 101], Chapter 7 in [99], Chapter 8 in [164], Chapter 9 in [36], Chapter 10 in [165], Chapter 11 in [73].

Bibliography

[1] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4):390–426, December 2003.

[2] S. K. Kaul, R. D. Yates, and M. Gruteser. Real-time status: How often should one update? In *IEEE Infocom*, March 2012.

[3] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano. Scheduling policies for minimizing age of information in broadcast wireless networks. *IEEE/ACM Transactions on Networking*, 26(6):2637–2650, December 2018.

[4] M. Costa, M. Codrenau, and A. Ephremides. Age of information with packet management. In *IEEE ISIT*, June 2014.

[5] A. M. Bedewy, Y. Sun, and N. B. Shroff. Optimizing data freshness, throughput, and delay in multi-server information-update systems. In *IEEE ISIT*, July 2016.

[6] Q. He, D. Yuan, and A. Ephremides. Optimizing freshness of information: On minimum age link scheduling in wireless systems. In *IEEE WiOpt*, May 2016.

[7] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides. Age of information with a packet deadline. In *IEEE ISIT*, July 2016.

[8] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff. Update or wait: How to keep your data fresh. *IEEE Transactions on Information Theory*, 63(11):7492–7508, November 2017.

[9] E. Najm and E. Telatar. Status updates in a multi-stream M/G/1/1 preemptive queue. In *IEEE Infocom*, April 2018.

[10] E. Najm, R. D. Yates, and E. Soljanin. Status updates through M/G/1/1 queues with HARQ. In *IEEE ISIT*, June 2017.

[11] A. Soysal and S. Ulukus. Age of information in G/G/1/1 systems. In *Asilomar Conference*, November 2019.

[12] A. Soysal and S. Ulukus. Age of information in G/G/1/1 systems: Age expressions, bounds, special cases, and optimization. *IEEE Trans. on Information Theory*, 2021. To appear. Available on arXiv: 1905.13743.

[13] B. Buyukates and S. Ulukus. Age of information with Gilbert-Elliot servers and samplers. In *CISS*, March 2020.

[14] R. D. Yates. The age of information in networks: Moments, distributions, and sampling. *IEEE Transactions on Information Theory*, 66(9):5712–5728, May 2020.

[15] R. Talak, S. Karaman, and E. Modiano. Minimizing age-of-information in multi-hop wireless networks. In *Allerton Conference*, October 2017.

[16] V. Tripathi and S. Moharir. Age of information in multi-source systems. In *IEEE Globecom*, December 2017.

[17] A. M. Bedewy, Y. Sun, and N. B. Shroff. Age-optimal information updates in multihop networks. In *IEEE ISIT*, June 2017.

[18] A. M. Bedewy, Y. Sun, and N. B. Shroff. The age of information in multihop networks. *IEEE/ACM Transactions on Networking*, 27(3):1248–1257, June 2019.

[19] J. Zhong, E. Soljanin, and R. D. Yates. Status updates through multicast networks. In *Allerton Conference*, October 2017.

[20] J. Zhong, R. D. Yates, and E. Soljanin. Multicast with prioritized delivery: How fresh is your data? In *IEEE SPAWC*, June 2018.

[21] B. Buyukates, A. Soysal, and S. Ulukus. Age of information in two-hop multicast networks. In *Asilomar Conference*, October 2018.

[22] B. Buyukates, A. Soysal, and S. Ulukus. Age of information in multihop multicast networks. *Journal of Communications and Networks*, 21(3):256–267, July 2019.

[23] K. S. A. Krishnan and V. Sharma. Minimizing age of information in a multihop wireless network. In *IEEE ICC*, June 2020.

[24] B. Buyukates, A. Soysal, and S. Ulukus. Age of information in multicast networks with multiple update streams. In *Asilomar Conference*, November 2019.

[25] S. Farazi, A. G. Klein, and D. R. Brown III. Fundamental bounds on the age of information in multi-hop global status update networks. *Journal of Communications and Networks, special issue on Age of Information*, 21(3):268–279, July 2019.

[26] S. Ioannidis, A. Chaintreau, and L. Massoulie. Optimal and scalable distribution of content updates over a mobile social network. In *IEEE Infocom*, April 2009.

[27] Y. Azar, E. Horvitz, E. Lubetzky, Y. Peres, and D. Shahaf. Tractable near-optimal policies for crawling. *PNAS*, 115(32):8099–8103, August 2018.

[28] A. Kolobov, Y. Peres, E. Lubetzky, and E. Horvitz. Optimal freshness crawl under politeness constraints. In *ACM SIGIR Conference*, July 2019.

[29] B. E. Brewington and G. Cybenko. Keeping up with the changing web. *Computer*, 33(5):52–58, May 2000.

[30] M. Wang, W. Chen, and A. Ephremides. Reconstruction of counting process in real-time: The freshness of information through queues. In *IEEE ICC*, July 2019.

[31] Y. Sun, Y. Polyanskiy, and E. Uysal-Biyikoglu. Remote estimation of the Wiener process over a channel with random delay. In *IEEE ISIT*, June 2017.

[32] Y. Sun and B. Cyr. Information aging through queues: A mutual information perspective. In *IEEE SPAWC*, June 2018.

[33] J. Chakravorty and A. Mahajan. Remote estimation over a packet-drop channel with Markovian state. *IEEE Transactions on Automatic Control*, 65(5):2016–2031, July 2020.

[34] C. Kam, S. Kompella, and A. Ephremides. Age of incorrect information for remote estimation of a binary Markov source. In *IEEE Infocom*, July 2020.

[35] A. Arafa, K. Banawan, K. G. Seddik, and H. V. Poor. Sample, quantize and encode: Timely estimation over noisy channels. *IEEE Transactions on Communications*, 2021. To appear. Available on arXiv:2007.10200.

[36] M. Bastopcu and S. Ulukus. Who should Google Scholar update more often? In *IEEE Infocom*, July 2020.

[37] B. T. Bacinoglu, E. T. Ceran, and E. Uysal-Biyikoglu. Age of information under energy replenishment constraints. In *UCSD ITA*, February 2015.

[38] B. T. Bacinoglu and E. Uysal-Biyikoglu. Scheduling status updates to minimize age of information with an energy harvesting sensor. In *IEEE ISIT*, June 2017.

[39] B. T. Bacinoglu, Y. Sun, E. Uysal-Biyikoglu, and V. Mutlu. Achieving the age-energy trade-off with a finite-battery energy harvesting source. In *IEEE ISIT*, June 2018.

[40] A. Baknina, O. Ozel, J. Yang, S. Ulukus, and A. Yener. Sending information through status updates. In *IEEE ISIT*, June 2018.

[41] A. Baknina and S. Ulukus. Coded status updates in an energy harvesting erasure channel. In *CISS*, March 2018.

[42] X. Wu, J. Yang, and J. Wu. Optimal status update for age of information minimization with an energy harvesting source. *IEEE Transactions on Green Communications and Networking*, 2(1):193–204, March 2018.

[43] S. Feng and J. Yang. Optimal status updating for an energy harvesting sensor with a noisy channel. In *IEEE Infocom*, April 2018.

[44] S. Feng and J. Yang. Minimizing age of information for an energy harvesting source with updating failures. In *IEEE ISIT*, June 2018.

[45] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor. Age-minimal online policies for energy harvesting sensors with incremental battery recharges. In *UCSD ITA*, February 2018.

[46] A. Arafa, J. Yang, and S. Ulukus. Age-minimal online policies for energy harvesting sensors with random battery recharges. In *IEEE ICC*, May 2018.

[47] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor. Age-minimal transmission for energy harvesting sensors with finite batteries: online policies. *IEEE Transactions on Information Theory*, 66(1):534–556, January 2020.

[48] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor. Online timely status updates with erasures for energy harvesting sensors. In *Allerton Conference*, October 2018.

[49] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor. Using erasure feedback for online timely updating with an energy harvesting sensor. In *IEEE ISIT*, July 2019.

[50] A. Arafa and S. Ulukus. Age minimization in energy harvesting communications: Energy-controlled delays. In *Asilomar Conference*, October 2017.

[51] A. Arafa and S. Ulukus. Age-minimal transmission in energy harvesting two-hop networks. In *IEEE Globecom*, December 2017.

[52] S. Farazi, A. G. Klein, and D. R. Brown III. Average age of information for status update systems with an energy harvesting server. In *IEEE Infocom*, April 2018.

[53] S. Leng and A. Yener. Age of information minimization for an energy harvesting cognitive radio. *IEEE Transactions on Cognitive Communications and Networking*, 5(2):427–439, June 2019.

[54] Z. Chen, N. Pappas, E. Bjornson, and E. G. Larsson. Age of information in a multiple access channel with heterogeneous traffic and an energy harvesting node. In *IEEE Infocom*, April 2019.

[55] A. Arafa and S. Ulukus. Timely updates in energy harvesting two-hop networks: Offline and online policies. *IEEE Transactions on Wireless Communications*, 18(8):4017–4030, August 2019.

[56] R. V. Bhat, R. Vaze, and M. Motani. Throughput maximization with an average age of information constraint in fading channels. *IEEE Transactions on Wireless Communications*, 20(1):481–494, January 2021.

[57] J. Ostman, R. Devassy, G. Durisi, and E. Uysal. Peak-age violation guarantees for the transmission of short packets over fading channels. In *IEEE Infocom*, April 2019.

[58] S. Nath, J. Wu, and J. Yang. Optimizing age-of-information and energy efficiency tradeoff for mobile pushing notifications. In *IEEE SPAWC*, July 2017.

[59] Y. Hsu. Age of information: Whittle index for scheduling stochastic arrivals. In *IEEE ISIT*, June 2018.

[60] I. Kadota, A. Sinha, and E. Modiano. Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints. *IEEE/ACM Transactions on Networking*, 27(4):1359–1372, August 2019.

[61] A. Kosta, N. Pappas, A. Ephremides, M. Kountouris, and V. Angelakis. Age and value of information: Non-linear age case. In *IEEE ISIT*, June 2017.

[62] M. Bastopcu and S. Ulukus. Age of information with soft updates. In *Allerton Conference*, October 2018.

[63] M. Bastopcu and S. Ulukus. Minimizing age of information with soft updates. *Journal of Communications and Networks, special issue on Age of Information*, 21(3):233–243, July 2019.

[64] B. Buyukates, A. Soysal, and S. Ulukus. Age of information scaling in large networks. In *IEEE ICC*, May 2019.

[65] B. Buyukates, A. Soysal, and S. Ulukus. Age of information scaling in large networks with hierarchical cooperation. In *IEEE Globecom*, December 2019.

[66] B. Buyukates, A. Soysal, and S. Ulukus. Scaling laws for age of information in wireless networks. *IEEE Transactions on Wireless Communications*, 20(4):2413–2427, April 2021.

[67] J. Zhong, R. D. Yates, and E. Soljanin. Minimizing content staleness in dynamo-style replicated storage systems. In *IEEE Infocom*, April 2018.

[68] N. Rajaraman, R. Vaze, and G. Reddy. Not just age but age and quality of information. *IEEE Journal on Selected Areas in Communications*, 39(5):1325–1338, May 2021.

[69] Z. Liu and B. Ji. Towards the tradeoff between service performance and information freshness. In *IEEE ICC*, May 2019.

[70] A. Maatouk, M. Assaad, and A. Ephremides. The age of incorrect information: an enabler of semantics-empowered communication. December 2020. Available on arXiv:2012.13214.

[71] E. Uysal, O. Kaya, A. Ephremides, J. Gross, M. Codreanu, P. Popovski, M. Assaad, G. Liva, A. Munari, T. Soleymani, B. Soret, and K. H. Johansson. Semantic communications in networked systems. March 2021. Available on arXiv:2103.05391.

[72] O. Ayan, M. Vilgelm, M. Kluge, S. Hirche, and W. Kellerer. Age-of-information vs. value-of-information scheduling for cellular networked control systems. In *ACM/IEEE ICCPS*, April 2019.

[73] M. Bastopcu and S. Ulukus. Timely group updating. In *CISS*, March 2021.

[74] R. D. Yates and S. K. Kaul. The age of information: Real-time status updating by multiple sources. *IEEE Transactions on Information Theory*, 65(3):1807–1827, March 2019.

[75] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff. Age-optimal sampling and transmission scheduling in multi-source systems. In *ACM MobiHoc*, July 2019.

[76] S. Banerjee, R. Bhattacharjee, and A. Sinha. Fundamental limits of age-of-information in stationary and non-stationary environments. In *IEEE ISIT*, June 2020.

[77] J. Zhong and R. D. Yates. Timeliness in lossless block coding. In *IEEE DCC*, March 2016.

[78] J. Zhong, R. D. Yates, and E. Soljanin. Timely lossless source coding for randomly arriving symbols. In *IEEE ITW*, November 2018.

[79] P. Mayekar, P. Parag, and H. Tyagi. Optimal lossless source codes for timely updates. In *IEEE ISIT*, June 2018.

[80] P. Mayekar, P. Parag, and H. Tyagi. Optimal source codes for timely updates. *IEEE Transactions on Information Theory*, 66(6):3714–3731, March 2020.

[81] M. Bastopcu, B. Buyukates, and S. Ulukus. Optimal selective encoding for timely updates. In *CISS*, March 2020.

[82] B. Buyukates, M. Bastopcu, and S. Ulukus. Optimal selective encoding for timely updates with empty symbol. In *IEEE ISIT*, June 2020.

[83] M. Bastopcu, B. Buyukates, and S. Ulukus. Selective encoding policies for maximizing information freshness. *IEEE Transactions on Communications*, 2021. To appear. Available on arXiv:2004.06091.

[84] D. Ramirez, E. Erkip, and H. V. Poor. Age of information with finite horizon and partial updates. In *IEEE ICASSP*, May 2020.

[85] A. Arafa, K. Banawan, K. G. Seddik, and H. V. Poor. On timely channel coding with hybrid ARQ. In *IEEE Globecom*, December 2019.

[86] A. Arafa and R. D. Wesel. Timely transmissions using optimized variable length coding. In *IEEE Infocom*, May 2021.

[87] M. Bastopcu and S. Ulukus. Partial updates: Losing information for freshness. In *IEEE ISIT*, June 2020.

[88] M. A. Abd-Elmagid and H. S. Dhillon. Average peak age-of-information minimization in UAV-assisted IoT networks. *IEEE Transactions on Vehicular Technology*, 68(2):2003–2008, February 2019.

[89] J. Liu, X. Wang, and H. Dai. Age-optimal trajectory planning for UAV-assisted data collection. In *IEEE Infocom*, April 2018.

[90] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon. On the role of age of information in the internet of things. *IEEE Communications Magazine*, 57(12):72–77, December 2019.

[91] A. Alabbasi and V. Aggarwal. Joint information freshness and completion time optimization for vehicular networks. *IEEE Transactions on Services Computing*, pages 1–14, March 2020.

[92] W. Gao, G. Cao, M. Srivatsa, and A. Iyengar. Distributed maintenance of cache freshness in opportunistic mobile networks. In *IEEE ICDCS*, June 2012.

[93] R. D. Yates, P. Ciblat, A. Yener, and M. Wigger. Age-optimal constrained cache updating. In *IEEE ISIT*, June 2017.

[94] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides. Information freshness and popularity in mobile caching. In *IEEE ISIT*, June 2017.

[95] S. Zhang, J. Li, H. Luo, J. Gao, L. Zhao, and X. S. Shen. Towards fresh and low-latency content delivery in vehicular networks: An edge caching aspect. In *IEEE WCSP*, October 2018.

[96] H. Tang, P. Ciblat, J. Wang, M. Wigger, and R. Yates. Age of information aware cache updating with file- and age-dependent update durations. In *IEEE WiOpt*, June 2020.

[97] J. Zhong, R. D. Yates, and E. Soljanin. Two freshness metrics for local cache refresh. In *IEEE ISIT*, June 2018.

[98] L. Yang, Y. Zhong, F. Zheng, and S. Jin. Edge caching with real-time guarantees. December 2019. Available on arXiv:1912.11847.

[99] M. Bastopcu and S. Ulukus. Maximizing information freshness in caching systems with limited cache storage capacity. In *Asilomar Conference*, November 2020.

[100] M. Bastopcu and S. Ulukus. Cache freshness in information updating systems. In *CISS*, March 2021.

[101] M. Bastopcu and S. Ulukus. Information freshness in cache updating systems. *IEEE Transactions on Wireless Communications*, 20(3):1861–1874, March 2021.

[102] P. Kaswan, M. Bastopcu, and S. Ulukus. Freshness based cache updating in parallel relay networks. In *IEEE ISIT*, July 2021.

[103] Y. Gu, Q. Wang, H. Chen, Y. Li, and B. Vucetic. Optimizing information freshness in two-hop status update systems under a resource constraint. *IEEE Journal on Selected Areas in Communications*, 39(5):1380–1392, May 2021.

[104] Q. Kuang, J. Gong, X. Chen, and X. Ma. Age-of-information for computation-intensive messages in mobile edge computing. January 2019. Available on arXiv: 1901.01854.

[105] J. Gong, Q. Kuang, X. Chen, and X. Ma. Reducing age-of-information for computation-intensive messages via packet replacement. In *IEEE WCSP*, October 2019.

[106] P. Zou, O. Ozel, and S. Subramaniam. Trading off computation with transmission in status update systems. In *IEEE PIMRC*, September 2019.

[107] A. Arafa, R. D. Yates, and H. V. Poor. Timely cloud computing: preemption and waiting. In *Allerton Conference*, September 2019.

[108] M. Bastopcu and S. Ulukus. Age of information for updates with distortion. In *IEEE ITW*, August 2019.

[109] M. Bastopcu and S. Ulukus. Age of information for updates with distortion: Constant and age-dependent distortion constraints. *IEEE/ACM Transactions on Networking*, 2021. To appear. Available on arXiv:1912.13493.

[110] A. Behrouzi-Far and E. Soljanin. On the effect of task-to-worker assignment in distributed computing systems with stragglers. In *Allerton Conference*, October 2018.

[111] B. Buyukates and S. Ulukus. Timely updates in distributed computation systems with stragglers. In *Asilomar Conference*, November 2020.

[112] B. Buyukates and S. Ulukus. Timely distributed computation with stragglers. *IEEE Transactions on Communications*, 68(9):5273–5282, September 2020.

[113] P. Zou, O. Ozel, and S. Subramaniam. Optimizing information freshness through computation–transmission tradeoff and queue management in edge computing. *IEEE/ACM Transactions on Networking*, 29(2):949–963, April 2021.

[114] E. T. Ceran, D. Gunduz, and A. Gyorgy. A reinforcement learning approach to age of information in multi-user networks. In *IEEE PIMRC*, September 2018.

[115] H. B. Beytur and E. Uysal-Biyikoglu. Age minimization of multiple flows using reinforcement learning. In *IEEE ICNC*, February 2019.

[116] M. A. Abd-Elmagid, H. S. Dhillon, and N. Pappas. A reinforcement learning framework for optimizing age of information in RF-powered communication systems. *IEEE Transactions on Communications*, 68(8):4747–4760, August 2020.

[117] H. Tang, J. Wang, L. Song, and J. Song. Scheduling to minimize age of information in multi-state time-varying networks with power constraints. In *Allerton Conference*, September 2019.

[118] J. Yun, C. Joo, and A. Eryilmaz. Optimal real-time monitoring of an information source under communication costs. In *IEEE CDC*, December 2018.

[119] G. Stamatakis, N. Pappas, A. Fragkiadakis, and A. Traganitis. Autonomous maintenance in IoT networks via AoI-driven deep reinforcement learning. In *IEEE Infocom*, May 2021.

[120] N. Zhang, J. Liu, L. Xie, and P. Tong. A deep reinforcement learning approach to energy-harvesting UAV-aided data collection. In *IEEE WCSP*, October 2020.

[121] X. Wu, X. Li, J. Li, P. C. Ching, and H. V. Poor. Deep reinforcement learning for iot networks: Age of information and energy cost tradeoff. In *IEEE Globecom*, December 2020.

[122] E. U. Atay, I. Kadota, and E. Modiano. Aging bandits: Regret analysis and order-optimal learning algorithm for wireless networks with stochastic arrivals. December 2020. Available on arXiv:2012.08682.

[123] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor. Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning. *IEEE Transactions on Communications*, 68(11):6807–6821, November 2020.

[124] S. Leng and A. Yener. Age of information minimization for wireless ad hoc networks: A deep reinforcement learning approach. In *IEEE Globecom*, December 2019.

[125] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad. Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks. In *IEEE Globecom*, December 2019.

[126] A. Elgabli, H. Khan, M. Krouka, and M. Bennis. Reinforcement learning based scheduling algorithm for optimizing age of information in ultra reliable low latency networks. In *IEEE ISCC*, June 2019.

[127] B. Buyukates and S. Ulukus. Timely communication in federated learning. In *IEEE Infocom*, May 2021.

[128] E. Ozfatura, B. Buyukates, D. Gunduz, and S. Ulukus. Age-based coded computation for bias reduction in distributed learning. In *IEEE Globecom*, December 2020.

[129] A. Kosta, N. Pappas, and V. Angelakis. Age of information: A new concept, metric, and tool. *Foundations and Trends in Networking*, 12(3):162–259, November 2017.

[130] R. D. Yates, Y. Sun, D. R. Brown III, S. K. Kaul, E. Modiano, and S. Ulukus. Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications*, 39(5):1183–1210, May 2021.

[131] R. D. Yates. The age of gossip in networks. In *IEEE ISIT*, July 2021.

[132] B Buyukates, M. Bastopcu, and S. Ulukus. Age of gossip in networks with community structure. In *IEEE SPAWC*, September 2021.

[133] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh. Fresh caching for dynamic content. In *IEEE Infocom*, May 2021.

[134] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides. The age of incorrect information: A new performance metric for status updates. *IEEE/ACM Transactions on Networking*, 28(5):2215–2228, July 2020.

[135] R. D. Yates and S. K. Kaul. Real-time status updating: Multiple sources. In *IEEE ISIT*, July 2012.

[136] S. K. Kaul, R. D. Yates, and M. Gruteser. Status updates through queues. In *CISS*, March 2012.

[137] C. Kam, S. Kompella, and A. Ephremides. Age of information under random updates. In *IEEE ISIT*, July 2013.

[138] C. Kam, S. Kompella, G. D. Nguyen, and J. E. Wieselthier. Towards an effective age of information: Remote estimation of a Markov source. In *IEEE Infocom*, April 2018.

[139] R. D. Yates, E. Najm, E. Soljanin, and J. Zhong. Timely updates over an erasure channel. In *IEEE ISIT*, June 2017.

[140] M. Bastopcu and S. Ulukus. Scheduling a human channel. In *Asilomar Conference*, October 2018.

[141] R. Singh, G. K. Kamath, and P. R. Kumar. Optimal information updating based on value of information. In *Allerton Conference*, September 2019.

[142] J. Ma, L. Liu, H. Song, and P. Fan. On the fundamental tradeoffs between video freshness and video quality in real-time applications. *IEEE Internet of Things Journal*, 8(3):1492–1503, February 2021.

[143] A. Arafa, K. Banawan, K. G. Seddik, and H. Vincent Poor. Timely estimation using coded quantized samples. In *IEEE ISIT*, June 2020.

[144] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[145] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[146] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 2012.

[147] R. D. Yates and D. J. Goodman. *Probability and Stochastic Processes*. Wiley, 2014.

[148] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):435–607, March 1967.

[149] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[150] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, December 1996.

[151] K. Ryan, A. Imen, P. Borne, and K. Mekki. Evolutionary Approach for the Containers Bin-Packing Problem. *Studies in Informatics and Control*, 18(4):315–324, November 2009.

[152] D. P. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs: Prentice Hall, 1989.

[153] J. A. O'Sullivan. Alternating minimization algorithms: From Blahut-Arimoto to expectation-maximization. *Codes, Curves and Signals:Common Threads in Communications*, 485:173–192, 1998.

[154] A. Yener, R. D. Yates, and S. Ulukus. Interference management for CDMA systems through power control, multiuser detection, and beamforming. *IEEE Transactions on Communications*, 49(7):1227–1239, July 2001.

[155] U. Niesen, D. Shah, and G. Wornell. Adaptive alternating minimization algorithms. In *IEEE ISIT*, June 2007.

[156] S. Zhang, N. Zhang, X. Fang, P. Yang, and X. S. Shen. Self-sustaining caching stations: toward cost-effective 5G-enabled vehicular networks. *IEEE Communications Magazine*, 55(11):202–208, November 2017.

[157] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *ACM SIGCOMM Computer Communication Review*, 32(4):177–190, August 2002.

[158] M. Wang, W. Chen, and A. Ephremides. Real-time reconstruction of a counting process through first-come-first-serve queue systems. *IEEE Transactions on Information Theory*, 66(7):4547–4562, July 2020.

[159] R. Dorfman. The detection of defective members of large populations. *Ann. Math. Statist.*, 14(4):436–440, December 1943.

[160] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella. Age-optimal updates of multiple information flows. In *IEEE Infocom*, April 2018.

[161] H. B. Beytur and E. Uysal-Biyikoglu. Minimizing age of information for multiple flows. In *IEEE BlackSeaCom*, June 2018.

[162] A. Maatouk, S. Kriouile, M. Assad, and A. Ephremides. On the optimality of the Whittle's index policy for minimizing the age of information. *IEEE Transactions on Wireless Communications*, 20(2):1263–1277, February 2021.

[163] O. T. Yavascan and E. Uysal. Analysis of slotted ALOHA with an age threshold. *IEEE Journal on Selected Areas in Communications*, 39(5):1456–1470, May 2021.

[164] M. Bastopcu, B. Buyukates, and S. Ulukus. Gossiping with binary freshness metric. July 2021. Available on arXiv:2107.14218.

[165] M. Bastopcu and S. Ulukus. Timely tracking of infection status of individuals in a population. In *IEEE Infocom*, May 2021.