OPEN-LOOP SCHEDULING FOR MINIMIZING POLYNOMIAL FUNCTIONS OF AGE OF INFORMATION

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

By

Mehmet Semih Saydam

July 2025

Open-Loop Scheduling for Minimizing Polynomial Functions of Age of Information
By Mehmet Semih Saydam
July 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Nail Akar(Advisor)

Ezhan Karaşan

Çağlar Tunç

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

OPEN-LOOP SCHEDULING FOR MINIMIZING POLYNOMIAL FUNCTIONS OF AGE OF INFORMATION

Mehmet Semih Saydam
M.S. in Electrical and Electronics Engineering
Advisor: Nail Akar
July 2025

Age of Information (AoI) is a metric that quantifies freshness of information in a status update system, making it crucial for applications where timely updates are essential, such as real-time monitoring systems, IoT networks, and mission-critical communication systems. This thesis explores age-agnostic cyclic scheduling in multi-source, single-server Generate-At-Will (GAW) status update systems, where the goal is to minimize the expected value of a non-linear polynomial function of AoI in a discrete-time setting. In the literature, approaches that aim to minimize the weighted sum of average AoI often rely solely on the first two moments of packet service times. However, the use of non-linear functions of age as the information freshness metric, requires an analytical model to obtain the distribution of AoI which also uses the distributions of packet service times as input to the model. In this work, given a cyclic transmission pattern, we use the theory of discrete-time absorbing Markov chains to obtain the distribution of the AoI of each user, which then allows us to find the expected value of any non-linear function of individual ages, referred to as the Value of Information (VoI) in this work. Subsequently, using the proposed analytical method, we propose a metaheuristic based space-search algorithm, specifically leveraging the Simulated Annealing (SA) technique, to obtain a cyclic schedule with the goal of minimizing a polynomial cost function of age. Numerical results are presented to validate the proposed approach.

Keywords: Age of Information, Functions of Age, multi-source status update systems, cyclic scheduling.

ÖZET

BİLGİ YAŞI FONKSİYONLARININ MİNİMİZASYONU

Mehmet Semih Saydam
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Nail Akar
Temmuz 2025

Bilgi Yaşı (AoI), bir durum güncelleme sistemindeki bilginin güncelliğini nicel olarak ifade eden bir ölçüttür. Bu ölçüt, gerçek zamanlı izleme sistemleri, Nesnelerin İnterneti (IoT) ağları ve görev-kritik haberleşme sistemleri gibi zamanında bilgi güncellemelerinin hayati önem taşıdığı uygulamalarda önemli bir rol oynar. Bu çalışma, çok kaynaklı ve tek sunuculu İsteğe-Bağlı-Üretim (GAW) modeline sahip durum güncelleme sistemlerinde, bilgi yaşından bağımsız çevrimsel zamanlama yaklaşımlarını incelemektedir. Temel hedef, ayrık zamanlı bir çerçevede bilgi yaşının doğrusal olmayan bir fonksiyonunun beklenen değerini en aza indirmeye çalışan bir çizelgeleme politikası geliştirmektir.

Bu amaç doğrultusunda, verilen bir döngüsel çizelgeleme desenine bağlı olarak her kullanıcının bilgi yaşı dağılımını elde etmek için ayrık zamanlı soğurucu Markov zincirleri teorisi kullanılmaktadır. Böylece, her bir kullanıcının bilgi yaşına uygulanan doğrusal olmayan bir maliyet fonksiyonunun beklenen değeri hesaplanabilmekte ve bu değer çalışmada, Bilgi Değeri (VoI) olarak adlandırılmaktadır. Son olarak, bilgi yaşının polinomsal bir maliyet fonksiyonu altında minimize edilmesini amaçlayan çevrimsel çizelgeleme deseni oluşturmak üzere meta-sezgisel bir arama algoritması geliştirilmiş ve yöntemin geçerliliği, sayısal sonuçlarla desteklenmiştir.

Anahtar sözcükler: Bilgi Yaşı, Yaş Fonksiyonları, Çok Kaynaklı Durum Güncelleme Sistemleri, Döngüsel Çizelgeleme.

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my thesis advisor Nail Akar for his invaluable guidance and unwavering support throughout the course of this work. His insights and encouragement have been instrumental in shaping my academic journey.

I am deeply thankful to my family, my mother Beyza, my father Numan, and my sister Elif, for their constant support and belief in me throughout my academic endeavors. Their presence and encouragement have been a source of strength in every step of this journey.

I would also like to extend my heartfelt thanks to my colleagues at ASELSAN for their friendship and collaboration, and especially to Dr. Ege Gamgam for sharing his valuable knowledge and perspectives with me.

I would like to acknowledge Utku Serhat Derici for his continuous encouragement and support, which have accompanied me from the beginning of my undergraduate education through the completion of this thesis.

In the later part of my academic journey, I had the chance to meet Sıla Seydim, whose companionship brought joy and balance to this challenging period. I am grateful for the positive impact she has had on my life.

Lastly, I want to thank all my friends who have supported me with their time, motivation, and encouragement during this process. Your help has meant more than words can express.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In recent years, the exponential growth in global connectivity and automation has significantly increased the importance of timely information delivery. Historically, traditional performance metrics such as throughput, delay, and reliability have been the primary tools for evaluating communication systems. However, with the increasing number of real-time and interactive applications, these conventional metrics have shown limitations in reflecting the freshness of information. This notion of freshness is represented by the Age of Information (AoI) metric, which models the staleness of data at the destination. For a given source, AoI is calculated as a stochastic process $\Delta(t)$, measuring the time that has elapsed since the most recently received update was generated. Mathematically, this is expressed as:

$$\Delta(t) = t - G(t), \tag{1.1}$$

where $G(t)$ denotes the generation time of the last successfully received status update. While AoI is classically defined in continuous time as $\Delta(t) = t - G(t)$, many practical systems-such as slotted networks and digital controllers operate in discrete time. In this thesis, we adopt a discrete-time model where time is

indexed by slots $k \in \mathbb{N}$ and age is denoted by Δ_k, which aligns better with slotted communication systems and supports discrete-time analytical tools used throughout this work. This discrete-time formulation offers a more accurate way to quantify freshness compared to traditional performance metrics. As an example, consider throughput, which measures the volume of data delivered per unit time, or delay, which tracks how long it takes for a message to travel from source to destination. While both remain important, they often fall short in capturing the freshness of information, especially in systems where decisions are made based on the most recently received data. A lower update rate may reduce congestion and improve average delay, but it risks leaving the destination with outdated information. Conversely, more frequent updates can enhance freshness but may overload the network, increase queuing delays, or cause packet drops due to limited bandwidth. These trade-offs illustrate that traditional metrics may conflict in dynamic systems and fail to provide a clear view of how current or useful the delivered information actually is [2].

Motivated by the limitations of traditional metrics, early research efforts proposed the AoI as a more nuanced indicator of data timeliness [3].

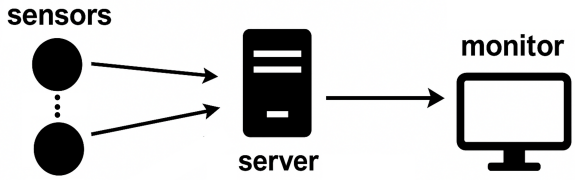

Figure 1.1: A multi-sensor, single-server update system.

Today, numerous data-driven applications depend on timely information exchange to ensure efficient monitoring, accurate control, and informed decision-making. Such applications span a broad spectrum of areas, including industrial automation systems, autonomous transportation networks, remote healthcare monitoring, smart city infrastructures, IoT systems, and military networks

involving target tracking, surveillance, and coordinated actions [4–15]. Together, these applications show that outdated information have less impact in value, leading to significant performance degradation, operational inefficiencies, and critical safety or security risks. Consequently, maintaining information freshness through effective AoI-aware system design and optimization has become an essential requirement for such applications. However, despite its effectiveness in capturing the timeliness of updates, the AoI metric assumes that all status updates carry equal significance, regardless of their actual content or contextual urgency. This assumption often fails to capture practical complexities in real-world applications where the operational value of an update can vary depending on the system state, mission objectives, or environmental context. For instance, in a military surveillance system or a medical monitoring setup, an update indicating a critical anomaly is far more valuable than a routine status report even if both are equally fresh.

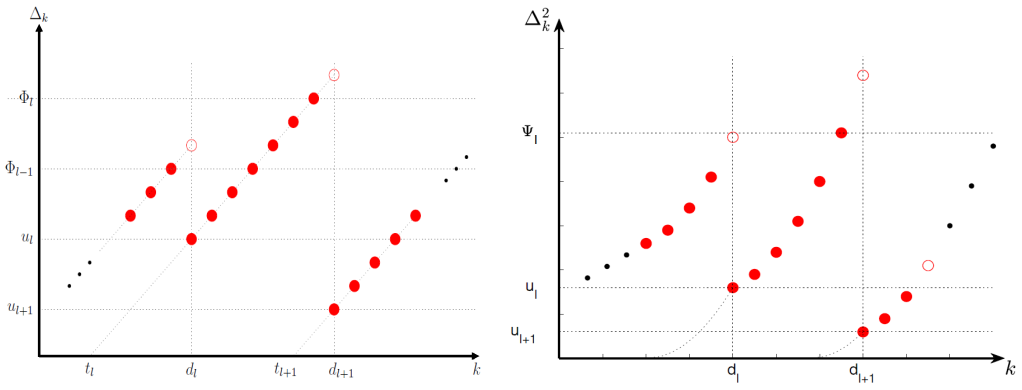

Figure 1.2: Sample path of the AoI process Δ_k (right) and function of AoI process $f(\Delta_k)$ (left)

To address this limitation, several alternative metrics have been proposed in the literature to better capture the relevance and utility of information. One such metric is the Value of Information (VoI) framework, which also serves as

the primary focus of this thesis and is often interpreted as a function of age throughout this work. VoI generalizes the AoI concept by incorporating content-awareness and decision impact into the evaluation of update utility [16]. Unlike AoI, which only accounts for the age of the last received data, VoI quantifies the expected benefit or cost reduction that a given update provides to the system upon arrival. This enables a more intelligent prioritization of updates by assigning source-specific, time-varying weights or costs that reflect their contribution to the system's overall performance or mission success [17, 18].

In the AoI literature, two main classes of schedulers are commonly distinguished: *closed-loop* and *open-loop* schedulers. Closed-loop schedulers adapt their decisions based on real-time observations of the service times and the AoI processes. Well-known examples include age-aware policies such as Max-Weight and Whittle index-based methods [19–21]. In contrast, open-loop schedulers operate in an age-agnostic manner, relying solely on prior knowledge of service time statistics. These schedulers can be either probabilistic-where sources are selected with predefined probabilities or cyclic, where a fixed transmission pattern is followed repeatedly [22]. To the best of our knowledge, these two represent the main families of open-loop schedulers that have been studied for GAW systems. Among them, probabilistic scheduling was shown to perform poorly in heterogeneous settings, as demonstrated in prior work [1]. Open-loop designs offer significant runtime efficiency, with cyclic policies, achieving constant-time decision complexity and eliminating the need for return channels or feedback mechanisms. In this thesis, we focus on open-loop cyclic scheduling strategies under the Generate-at-Will (GAW) model. This open-loop formulation not only reduces scheduling complexity but also ensures that each user knows its exact transmission slot in advance. Such deterministic access simplifies implementation in resource-constrained systems and aligns naturally with the absorbing Markov chain framework developed in this work. To fully leverage the benefits of VoI-aware system design, it is essential not only to define the value of updates but also to develop scheduling and optimization strategies that minimize the overall cost induced by stale information. While prior work has extensively explored the minimization of weighted average AoI which only requires first-moment age statistics our study takes a

fundamentally different path by targeting weighted average VoI. Since VoI is defined as a non-linear polynomial function over the age distribution, this shift necessitates a complete probabilistic characterization of each user's age process. Unlike existing closed-loop schemes that require real-time feedback or contention-based access, our open-loop design relies on a fixed cyclic schedule where each user knows its transmission slots, eliminating the need for return channels or coordination overhead. Consequently, our methodology departs from traditional AoI frameworks and constructs an exact distributional model using a discrete-time absorbing Markov chain. This choice of discrete time aligns naturally with slotted communication systems and enables tractable, analytical computation of user-specific performance. The proposed modeling for distributional analysis not just first-moment statistics marks a fundamental divergence from classical AoI scheduling approaches and constitutes the core novelty of this thesis, enabling accurate, application-aware optimization of scheduling policies that account for both content relevance and delivery timeliness.

1.2 Thesis Contribution

A key novelty of this work is the integration of the functions of age framework, also known as the VoI perspective. While existing literature has extensively studied scheduling policies for minimizing the weighted average AoI, these approaches typically focus on the expected age and do not account for content relevance or user-specific urgency. In contrast, our framework operates in the VoI domain, where cost depends non-linearly on the full age distribution. This requires an exact probabilistic modeling of the age process, fundamentally distinguishing our methodology from previous AoI-based studies. While the VoI concept has been previously investigated in [17, 23, 24], where its performance implications and advantages were analyzed across various system models, those studies did not propose a method for minimizing such cost functions. In contrast, our work develops a full framework for scheduling in discrete-time systems that directly targets the minimization of user-specific, non-linear age-based cost functions under GAW model. This enables the use of heterogeneous cost profiles that capture

the perceived importance or utility of timely updates from each source. As a result, the scheduler can be tailored to application-specific requirements, offering enhanced performance in diverse environments. Notably, when the cost function grows linearly with age, the proposed model reduces to the classical AoI formulation, making AoI a special case of our framework.

Another distinguishing aspect of this study is the use of the service time distribution for each user, rather than only relying on its statistical moments (e.g., mean or variance). This allows the system model to capture more accurate representation of service variability.

To compute the expected cost of a given cyclic schedule, we model the evolution of the system using a discrete-time absorbing Markov chain. This approach facilitates exact computation of the long-run average system value or age under any scheduling patterns, while avoiding the complexity of simulation-based estimation.

The scheduling problem in general heterogeneous systems is known to be NP-hard, meaning that finding the exact optimal solution becomes computationally infeasible as the system size grows. Even with a small number of users, the number of possible scheduling patterns increases exponentially with the pattern length, making exhaustive search impractical. This complexity becomes even more significant in our setting, where the cost depends on non-linear, user-specific VoI functions and full service time distributions. These factors make the cost landscape highly irregular and difficult to predict. As a result, simple or greedy approaches often get stuck in suboptimal solutions.

For the construction of scheduling patterns, we depart from the greedy search approach employed in [1] and instead utilize a metaheuristic algorithm based on Simulated Annealing (SA). The use of SA allows for more comprehensive exploration of the scheduling space, reducing the likelihood of convergence to suboptimal local minima and offering improved schedule quality in practice. To illustrate the structure of our approach, a high-level overview of the scheduling framework is provided in Figure 1.3.

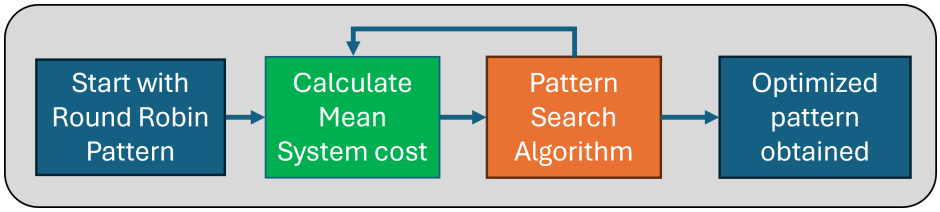

Figure 1.3: High-level flow of the pattern search framework, starting from a baseline schedule and iteratively optimizing system cost through a metaheuristic search algorithm.

The overall flow of the proposed scheduling framework is illustrated in Figure 1.3. The process begins with a simple round-robin pattern as the initial schedule. Using the absorbing Markov chain-based analytical method, the mean system cost is calculated for this pattern. A metaheuristic search algorithm specifically Simulated Annealing is then applied to explore the space of cyclic patterns and iteratively identify improved schedules. At each step, the system evaluates whether the new pattern leads to a lower cost. If so, it is retained and used as a base for further exploration. The process continues until a termination condition, such as a maximum number of iterations, is met. The final output is a scheduling pattern that, guided by the search algorithm's capability, achieves a reduced system cost. The cost evaluation procedure and the pattern construction algorithm are discussed in detail in the subsequent chapters.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Section 2 reviews related work on AoI, including scheduling strategies and recent extensions involving VoI. Section 3 introduces key mathematical tools, including absorbing Markov chains.

Section 4 defines the discrete-time system model and presents the VoI-based cost formulation. Section 5 details the analytical method for evaluating the expected cost of cyclic schedules using absorbing Markov chains.

Section 6 introduces a simulated annealing-based heuristic for scheduling pattern optimization. Numerical results are provided in Section 7, followed by concluding remarks and future directions in Section 8.

Chapter 2

Related Work

Most of the freshness metrics commonly used in the recent literature are based on the concept of AoI [24, 25], which quantifies the time elapsed since the generation of the most recently received status update at the destination (monitor) from a particular information source. A related metric, the Peak Age of Information (PAoI), is obtained by sampling the AoI process immediately before each packet reception event simulating worst-case freshness [26]. Depending on the specific application and system requirements, either AoI or PAoI is targeted for minimization. A wide range of studies in the literature have proposed methods for optimizing such metrics under various system models [25]. One of the earliest and most influential works in this area is presented in [2], which addresses a single-source status update system. Subsequent works have extended this model to incorporate queuing effects, different arrival processes, physical layer constraints, and lossy communication channels. A particularly relevant line of research focuses on systems with multiple information sources, which is also the setting considered in this thesis. In such multi-source scenarios, several system architectures have been studied [1, 27–31]. Status updates in these systems are typically modeled in two distinct forms. In the first, a central server actively polls the sources, prompting them to transmit updates in response. These updates often contain mission-critical information and require timely delivery. In the second model, known as the random arrival model, sources independently sample and transmit

updates at stochastic time instances without waiting for a poll message [29, 32]. These distinctions significantly influence the design of scheduling and resource allocation strategies aimed at minimizing AoI-related performance measures in distributed communication systems.

The frequency of packet generation and the queuing discipline adopted at the source significantly influence the evolution of the Age of Information, regardless of whether the system follows a poll-based or random arrival model. Several queuing strategies have been studied in the literature, including First-Come First-Served (FCFS), Last-Come First-Served with and without preemption (LCFS-P/NP), as well as various drop policies involving buffer replacement. Each of these approaches has distinct implications for AoI performance [2, 33].

In this work, we specifically focus on the GAW model, where sources are capable of generating packets at arbitrary time instances but are constrained to retain only the most recent status update. A similar setting has been considered in [1] within a continuous-time framework. However, our study differs in that we adopt a discrete-time model, which leads to fundamental changes in the system dynamics and performance analysis.

Another important observation concerns how AoI is computed in many of the earlier works. Typically, age is modeled as the time elapsed since the generation of the last received packet, increasing linearly between receptions and resetting upon update delivery. While this characterization is accurate and analytically convenient, it may not be adequate for all application contexts. To address such limitations, several extensions and alternative age-based metrics have been proposed in the literature to capture application-specific notions of freshness.

One such alternative metric is the Age of Incorrect Information (AoII), which considers not only the age of the last received update but also how accurately it reflects the current state of a dynamically evolving process. In systems where the state changes over time such as tracking mobile entities or monitoring stochastic processes the usefulness of an update may degrade in a manner that depends on the divergence between the actual state and the receiver's estimate, rather than

simply on elapsed time. AoII captures this by quantifying the estimation error between the true system state and the stale information held at the receiver [34–37]. An example application of AoII is presented in slotted ALOHA systems [38].

Beyond AoII, the literature also introduces other alternative freshness metrics, including the Effective Age [39] and the Peak Age Violation [40], each capturing different aspects of information timeliness, update relevance, or system reliability.

Among these diverse metrics, the main focus of this work is the Value of Information framework. VoI extends the concept of freshness by applying non-linear cost functions that reflect the urgency or contextual importance of individual updates [23, 41]. In contrast to AoI, which treats all packets equally regardless of their semantic value, VoI enables differentiation among updates by penalizing outdated information more heavily when it is considered more critical to the application.

Earlier works have analyzed VoI in systems using service disciplines such as FCFS and LCFS, demonstrating its benefits through mean-value performance metrics [18]. However, these studies are mostly analytical and focus on characterizing the behavior of VoI under fixed policies.

In contrast, while absorbing Markov chain models have been employed in prior works [42, 43], our study integrates this analytical tool within an optimization-based framework specifically designed to minimize the Value of Information in discrete-time systems. By leveraging this framework, we are able to evaluate the long-term performance of scheduling patterns in terms of their informational impact. Building upon the foundations laid in [1], where cyclic scheduling methods were introduced for discrete-time systems, our approach extends these ideas by jointly combining theoretical analysis and scheduling optimization.

Chapter 3

Preliminaries

3.1 AoI Definition

In discrete-time status update systems, AoI is a metric that quantifies the staleness of the most recently received data. Let Δ_k denote the instantaneous AoI observed at the beginning of time slot k. If the most recently received update was generated at time t_{latest}, then the AoI at time k is defined as:

$$\Delta_k = k - t_{\text{latest}} \tag{3.1}$$

Between two successive receptions of updates from the same source, Δ_k increases linearly with time. Figure 3.1 depicts the sample evolution of the discrete-time AoI and PAoI processes over time.

During each update cycle l, the instantaneous AoI, Δ_k, increases linearly over time, starting from $u_l = d_l - t_l$ at the reception time d_l, and growing incrementally with each time slot until the next update arrives. This growth reflects the accumulating staleness of information in the absence of fresh updates. The AoI attains its peak at the final slot of the update cycle, specifically at $k = d_{l+1} - 1$,

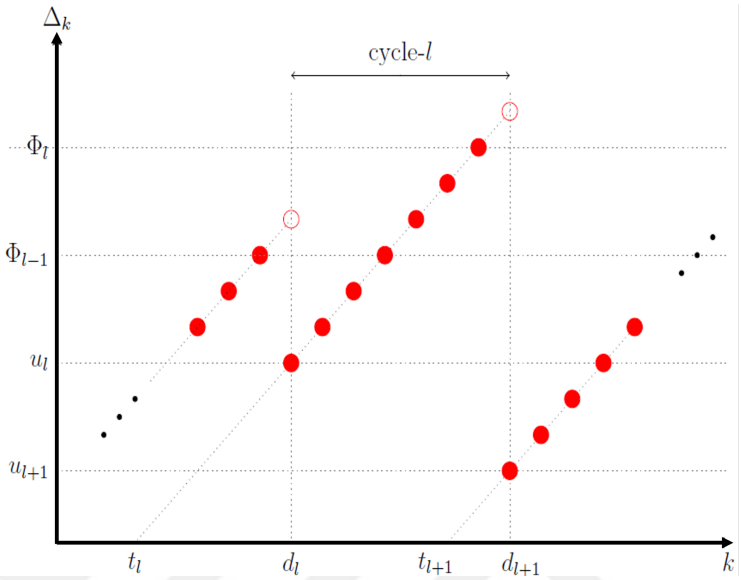

Figure 3.1: Sample path of the discrete-time AoI and PAoI processes for a user. The filled red circles represent time slots that contribute to the Age of Information, while the hollow red circles indicate the Peak Age of Information values at successful delivery times. .

right before the next update is received. Upon successful reception of a new update at d_{l+1}, the process resets to u_{l+1}, the system time of the next packet. A hollow red circle marks a hypothetical overshoot value representing the AoI had the update been delayed by one more slot.

Let Δ and Φ denote the corresponding steady-state random variables for AoI and PAoI processes. Their long-term expectations are computed as time-averages due to the ergodicity of the system:

$$\mathbb{E}[\Delta] = \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} \Delta_k \tag{3.2}$$

$$\mathbb{E}[\Phi] = \lim_{L\to\infty} \frac{1}{L} \sum_{l=1}^{L} \Phi_l \tag{3.3}$$

We further define the system-level AoI and PAoI as weighted averages across all

users:

$$\mathbb{E}[\Delta_{\text{sys}}] = \sum_{i=1}^{N} \omega_i \, \mathbb{E}[\Delta_i] \tag{3.4}$$

$$\mathbb{E}[\Phi_{\text{sys}}] = \sum_{i=1}^{N} \omega_i \, \mathbb{E}[\Phi_i] \tag{3.5}$$

where ω_i is the normalized weight of user i, with $\sum_i \omega_i = 1$, reflecting the relative importance or urgency of its updates.

3.2 VoI Definition

In status update systems, AoI is a widely used metric that measures the time elapsed since the most recently received update was generated. At any slot k, the AoI was previously defined in equation (3.1). Now, we define the VoI as a function of AoI, at each slot k, VoI is computed as:

$$V_k = f(\Delta_k) \tag{3.6}$$

where $f : \mathbb{N} \to \mathbb{R}^+$ is a polynomial cost function. Polynomial functions are adopted in this work as the mapping from AoI to VoI due to their analytical closed-form properties. Specifically, polynomial cost functions allow for exact expressions in the evaluation of expected cost, which is essential for the Markov chain-based analysis framework employed throughout this study. Among various options, we focus on second-order polynomial functions to maintain a balance between expressive power and mathematical simplicity. However, one can select a polynomial of arbitrary order N and apply the same framework. While the approach can also be extended to more general, non-polynomial cost functions, such extensions are beyond the scope of this work.

Similar to AoI, we are interested in both the instantaneous and peak values of VoI. Let V denote the steady-state distribution of V_k, and Ψ denote the cycle-wise peak VoI defined as $\Psi_l = f(\Phi_l)$, where Φ_l is the peak AoI during the lth update

cycle. Their distributions are given by:

$$\begin{aligned} p_V(v) &= \lim_{k\to\infty} \Pr(V_k = v), \\ p_\Psi(v) &= \lim_{l\to\infty} \Pr(\Psi_l = v) \end{aligned} \tag{3.7}$$

To visualize the nonlinear nature of VoI, Fig. 3.2 illustrates a sample path of the VoI process where the cost function is chosen as $f(\Delta) = \Delta^2$.

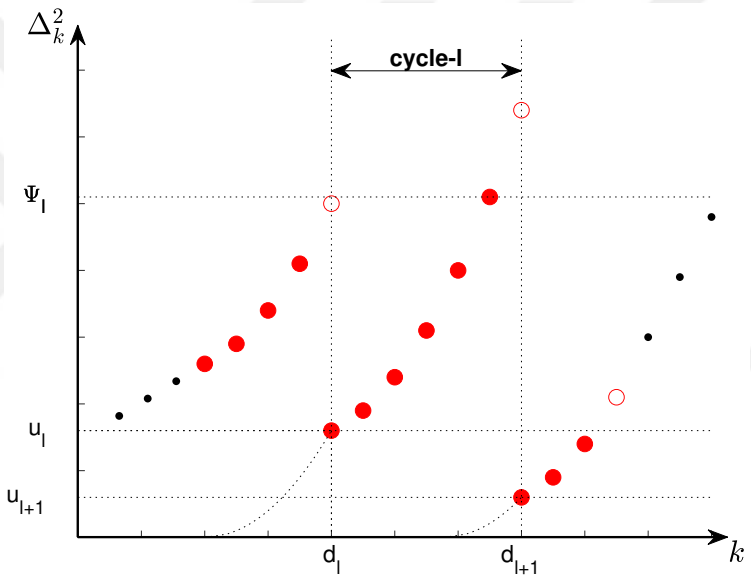

Figure 3.2: Sample path of the VoI process $f(\Delta_k) = \Delta_k^2$ under a quadratic cost function. The filled red circles represent time slots that contribute to the total Value of Information, corresponding to the instantaneous squared age values $f(\Delta_k) = \Delta_k^2$. The hollow red circles denote the Peak Value of Information values observed at successful update delivery times

During each update cycle l, the instantaneous VoI, $V_k = f(\Delta_k)$, grows quadratically over time, starting from u_l at the reception time d_l, and reaching a peak of $\Psi_l = f(\Phi_l)$. This sharp increase reflects the growing penalty for staleness in delay-sensitive applications. Upon successful reception of the next update at d_{l+1}, the process resets to u_{l+1}. A hollow red circle marks the hypothetical overshoot value, corresponding to the cost if the update were delayed by one more slot. Compared to the linear AoI process, this parabolic trajectory emphasizes the urgency of timely updates under nonlinear cost metrics.

3.3 Discrete-Time Absorbing Markov Chains

A discrete-time Markov chain (DTMC) is a stochastic process $\{X_k\}_{k\geq 0}$ defined on a finite or countable state space $\mathcal{X}$, such that the probability of transitioning to the next state depends only on the current state:

$$\mathbb{P}[X_{k+1} = j \mid X_k = i, X_{k-1}, \ldots, X_0] = \mathbb{P}[X_{k+1} = j \mid X_k = i] \tag{3.8}$$

The process evolves in discrete time steps, and its behavior is fully described by a transition probability matrix $P \in \mathbb{R}^{n\times n}$, where $P(i,j) = \mathbb{P}[X_{k+1} = j \mid X_k = i]$. A DTMC is said to be *absorbing* if there exists at least one state $s \in \mathcal{X}$ such that $\mathbb{P}[X_{k+1} = s \mid X_k = s] = 1$, meaning the process remains in state s once entered. Such a state is called an absorbing state. A Markov chain is considered absorbing if, starting from any transient state, there is a nonzero probability of eventually reaching some absorbing state. In other words, all states in the chain are either absorbing or transient, and every transient state can reach at least one absorbing state through a finite sequence of transitions with positive probability. In general, Markov chains may include multiple absorbing states, each potentially representing a distinct terminal outcome. However, in the analytical framework adopted in this work, each Markov chain is constructed such that all sample paths eventually lead to a single absorbing state. This modeling choice reflects the structure of our system, where the process terminates deterministically at a common endpoint, regardless of the path taken through the transient state space.

For an absorbing DTMC with r transient states and m absorbing states, the transition matrix P can be written in canonical form:

$$P = \begin{bmatrix} Q & R \\ \mathbf{0} & I \end{bmatrix} \tag{3.9}$$

where:

- $Q \in \mathbb{R}^{r\times r}$ contains the probabilities of transitioning among transient states,

- $R \in \mathbb{R}^{r \times m}$ contains the probabilities of transitioning from transient to absorbing states,
- $I \in \mathbb{R}^{m \times m}$ is the identity matrix over the absorbing states,
- $\mathbf{0}$ is a zero matrix of appropriate size.

The key to analyzing the behavior of an absorbing Markov chain lies in understanding how the system evolves before reaching an absorbing state. This is captured by the *fundamental matrix* $N = (I - Q)^{-1}$, where Q describes transitions among transient states. Each entry $N(i, j)$ represents the expected number of times the process visits state j starting from state i before absorption occurs. Using the fundamental matrix, one can also compute the expected time to absorption for each transient state. Specifically, the expected number of steps until absorption when starting from transient state i is given by

$$t(i) = \sum_{j=1}^{r} N(i, j),$$

which corresponds to the total expected number of visits to all transient states before the process is absorbed. Moreover, the absorption probabilities can be obtained using the matrix $B = NR$, where $B(i, j)$ gives the probability that the process is eventually absorbed in absorbing state j when starting from transient state i. Together, these quantities provide a complete description of the transient behavior and long-term outcomes in absorbing discrete-time Markov chains.

3.4 Discrete Phase-Type (DPH) Distributions

A discrete phase-type (DPH) distribution models the time until absorption in a finite discrete-time Markov chain (DTMC) with one absorbing state and a finite number of transient states. It provides a flexible way to represent a wide class of service time distributions including geometric, uniform, deterministic, and mixtures of such distributions as special cases.

Formally, let X be a discrete random variable denoting the number of time steps until absorption. Then, X is said to follow a DPH distribution, denoted by $X \sim \mathrm{DPH}(\boldsymbol{\alpha}, T)$, where:

- $\boldsymbol{\alpha} \in \mathbb{R}^{1\times r}$ is the initial probability vector over the r transient states,
- $T \in \mathbb{R}^{r\times r}$ is a substochastic transition matrix (i.e., each row sums to less than or equal to 1),
- $t = (I - T)\mathbf{1}$ is the absorption probability vector,

The probability mass function (PMF) of X is given by:

$$\mathbb{P}[X = k] = \boldsymbol{\alpha} T^{k-1} t, \quad k = 1, 2, \ldots \tag{3.10}$$

The DPH representation is typically structured in an upper-triangular form for canonical constructions. For a distribution with finite support $\{1, 2, \ldots, r\}$, the canonical transition matrix T may take the form:

$$T(i, i+1) = 1 - \frac{p_X(i)}{G_X(i)}, \quad \text{for } 1 \le i < r \tag{3.11}$$

where $G_X(i) = \sum_{j=i}^{r} p_X(j)$ is the tail probability at i.

$$T = \begin{bmatrix} 0 & T(1,2) & 0 & \cdots & 0 \\ 0 & 0 & T(2,3) & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & T(r-1,r) \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{3.12}$$

and the absorption vector is computed as:

$$t = (I - T)\mathbf{1} \tag{3.13}$$

DPH distributions are especially useful in queuing and network models where modeling non-memoryless service or delay processes is essential. They offer tractability while accommodating generality through matrix-based representations.

Chapter 4

System Model

We consider a slotted-time status update system composed of a single base station and N independent information sources, indexed by $n = 1, 2, \ldots, N$. Time is discretized into uniform-length slots, and the system evolves over these slots.

Each source periodically generates timestamped update packets destined for the base station. We consider GAW model, the scheduler actively polls users to generate a fresh update at the moment of scheduling.

Each source maintains a single-packet buffer, storing only the most recent packet. If a new packet arrives before the previous one is served, it replaces the older one in a preemptive manner. This ensures that the freshest information is always retained for transmission. Packet losses are assumed to be negligible; the system operates under ideal conditions without ARQ or HARQ mechanisms. The only source of delay is the service time, and potential control-plane latencies (e.g., polling commands) are considered negligible and thus excluded from the model.

Service is handled by a single server capable of serving at most one source at any time. Once a packet is selected for service, the server remains busy for a random duration S_n, representing the service time for source n. The service time may follow any discrete phase-type (DPH) distribution, independent across

sources and slots.

Scheduling decisions are made according to a fixed cyclic pattern $P = [P_0, P_1, \ldots, P_{K-1}]$ of length $K \geq N$, where each $P_k \in \{1, 2, \ldots, N\}$. Every source appears at least once in each cycle to prevent its age from diverging. At the start of a scheduling instant, the scheduler selects source P_k. If the source has a packet available, it is served for S_{P_k} time slots. Upon completion, the scheduler advances to $P_{(k+1) \bmod K}$ and the process repeats.

System performance is evaluated not directly in terms of raw age, but through a user-specific cost-of-age function. Let $\Delta_n(t)$ denote the AoI for source n at slot t, defined as the time elapsed since the most recently received update. The instantaneous VoI for user n is defined as:

$$V_n(t) = f_n(\Delta_n(t)), \tag{4.1}$$

where $f_n(\cdot)$ is typically a polynomial function capturing the cost of staleness for user n and the effective cost behavior is reflected by the first and second order terms, constant term is included for convenience and generality of the expression.

$$f_n(\Delta) = a_n\Delta^2 + b_n\Delta + c_n \tag{4.2}$$

In steady state, we consider the time-average VoI V_n and assign a weight $w_n > 0$ to reflect the priority or urgency of each source. The overall system cost is then expressed as a weighted sum:

$$\mathbb{E}[V] = \sum_{n=1}^{N} w_n \, \mathbb{E}[V_n], \tag{4.3}$$

which serves as the main performance metric to be minimized under the chosen scheduling pattern.

Chapter 5

Analysis of VoI with Absorbing Discrete Markov Chains

In this chapter, we present an analytical framework for evaluating the mean system VoI under a fixed cyclic scheduling pattern in a discrete-time setting. While the classical AoI formulation assumes a linear penalty that grows with time, our approach generalizes this notion by introducing a nonlinear cost model. Specifically, we allow the age to be mapped through user-defined functions, such as a quadratic cost function that enables differentiated prioritization based on the urgency or importance of updates from different sources:

$$f_i(\Delta) = a_i\Delta^2 + b_i\Delta + c_i \tag{5.1}$$

Our work extends existing studies that primarily analyze cyclic schedulers in continuous-time systems using only the first two moments (mean and variance) of service times. In contrast, we operate in discrete time, which more accurately reflects the operation of practical slotted communication systems, and we utilize the distribution of service times in our analysis.

The primary system-level objective in this framework is to minimize the expected overall cost induced by information staleness, quantified via the VoI model. This optimization problem is expressed as:

$$\min_{P} \quad J(P) = \sum_{i=1}^{N} \omega_i \, \mathbb{E}[f_i(\Delta_i)] \tag{5.2}$$

where:

- $P = [P_0, P_1, \ldots, P_{L-1}]$ is the cyclic scheduling pattern of length L,
- Δ_i is the random variable representing the age of user i,
- $f_i(\cdot)$ is the non-decreasing cost function (e.g., quadratic) that maps AoI to VoI,
- ω_i is the normalized weight that reflects the urgency of user i.

To evaluate this cost function under a given schedule, we construct a discrete-time absorbing Markov chain (DTMC) for each user. These chains are designed to capture the age evolution dynamics over repeated update cycles within the schedule. Each state transition in the chain represents a time slot, with transitions driven by the user's specific service success probabilities.

Given a fixed cyclic pattern $P = [P_0, P_1, \ldots, P_{K-1}]$, the sequence of scheduled users repeats deterministically over time. That is, the transmission order is predetermined and cyclically repeated across scheduling rounds. This structure allows us to analyze the long-term behavior of the system by focusing on the repeating pattern P, rather than considering the entire infinite time horizon.

To evaluate the VoI-based performance of such a system, we proceed in several steps. First, for each user i, we identify the subpatterns segments of P that start and end at two successive appearances of user i. These subpatterns represent the inter-update intervals for user i, during which its age evolves as other users are served. This step is described in detail in Section 5.1.

In the next step, we model the age dynamics of each user over its subpatterns using a DTMC. The states of this DTMC correspond to the positions in the subpattern, and transitions capture the stochastic service success behavior of users. The construction of the DTMC is explained in Section 5.2 for geometric service times. For DTMCs with more general discrete phase-type service models, the details are explained in Section 5.4.

After building the DTMC for each user, we compute the moments of the AoI distribution using the transient dynamics of the Markov chain. These moments are then used to evaluate the user-specific cost $\mathbb{E}[f_i(\Delta_i)]$ by applying the corresponding cost functions. The moment-based expectation calculation is detailed in Section 5.3.

Finally, the system-level VoI objective is obtained by aggregating the weighted costs across all users as in Equation (5.2). This modular framework allows for precise evaluation of user-defined cost functions under deterministic cyclic scheduling, accommodating heterogeneous service models and weights.

The remainder of this chapter details the formulation and solution of these Markov models and describes how they are used to compute the mean system VoI under a given cyclic schedule.

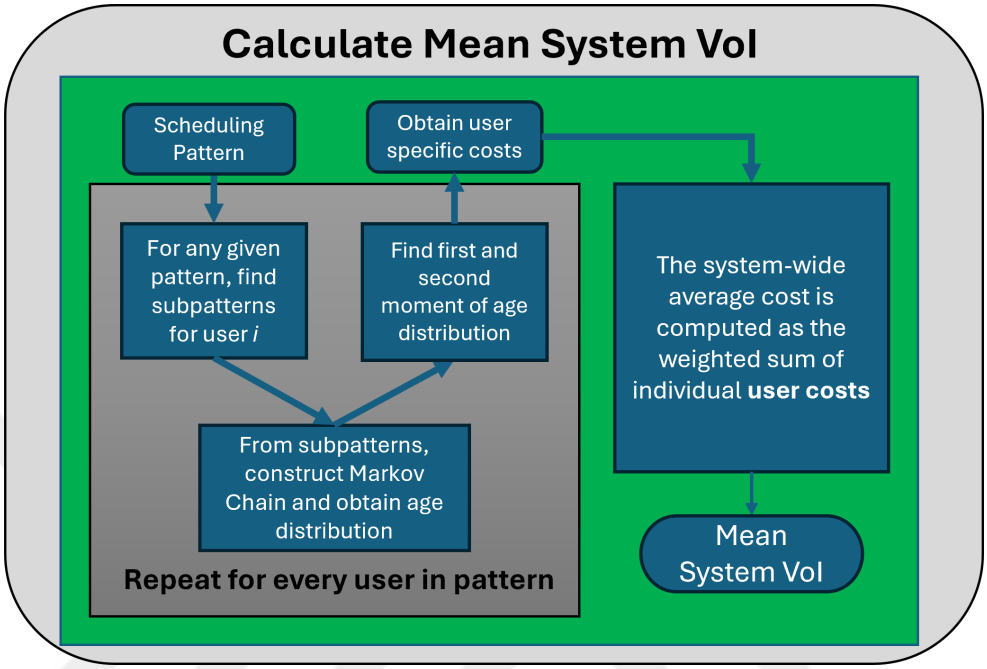

Figure 5.1: Overview of the cost evaluation process using absorbing Markov chains under a fixed cyclic scheduling pattern. The framework involves subpattern extraction, Markov chain construction, moment computation, and cost aggregation.

Figure 5.1 summarizes the complete evaluation pipeline described in this chapter. Starting from a cyclic scheduling pattern, the framework identifies user-specific subpatterns and constructs corresponding discrete-time Markov chains. From these chains, we extract the transition matrix and compute the first and second moments of the AoI distribution for each user. These moments are then mapped through user-defined cost functions such as the quadratic form in Equation (5.1) to evaluate the expected VoI cost. Finally, the system-wide performance is obtained as the weighted sum of individual user costs.

5.1 Subpattern Extraction

In a cyclic scheduling setup, a user may appear multiple times within a single scheduling cycle. However, each appearance occurs within a different context

preceded and followed by different sequences of other user transmissions. To accurately model the evolution of each user's information age, it is essential to extract and analyze the individual segments of the pattern that represent the user's inter-update intervals. We refer to these as subpatterns.

The purpose of subpattern extraction is twofold. First, it allows us to determine how many times each user is scheduled within one cycle of the pattern. Second, it enables the identification of the exact sequence of other users that are served between any two consecutive updates of a particular user. Since the user's age resets upon a successful transmission and increases during the intervening period, the structure of these subpatterns is crucial in capturing the age dynamics under fixed cyclic scheduling.

Let $P = [p_1, p_2, \ldots, p_L]$ denote the cyclic scheduling pattern of length L. For a given user i, we define a subpattern as a contiguous subsequence of P that starts and ends with i, and contains no other occurrence of i in between. Due to the cyclic nature of P, wrap-around logic is applied when identifying subpatterns that span the end and beginning of the pattern vector.

Example: Consider the cyclic pattern $P = [1\ 2\ 3\ 1\ 2]$. To extract the subpatterns for user 1:

- The first occurrence of user 1 is at position 1, and the next is at position 4. The users served in between are [2, 3], so the corresponding subpattern is:

$$\text{subpattern}_1 = [1\ 2\ 3\ 1]$$

- The second subpattern wraps around cyclically from index 4 to index 1, and includes user 2 again in between:

$$\text{subpattern}_2 = [1\ 2\ 1]$$

These subpatterns correspond to two distinct inter-update intervals for user 1, each with different users contributing to the waiting time between updates. Such

structural variations affect the user's age accumulation and, consequently, their contribution to the overall system cost.

One can extract the subpatterns for any user from any given cyclic pattern using the following pseudocode:

Algorithm 1 Extract Subpatterns for a Given User

Input: Cyclic pattern $P = [p_1, p_2, \ldots, p_L]$, user index u
Output: List of subpatterns $\mathcal{S}_u$
Initialize empty list $\mathcal{S}_u \leftarrow [\,]$
Let $I \leftarrow$ all positions in P where $P[i] = u$
for $j = 1$ to length of I **do**
    $start \leftarrow I[j]$
    $end \leftarrow I[(j+1) \bmod \text{length}(I)]$ ▷ Wraps cyclically
    if $end > start$ **then**
        $subpattern \leftarrow P[start : end]$
    else
        $subpattern \leftarrow P[start : L]$ concatenated with $P[1 : end]$
    end if
    Append $subpattern$ to $\mathcal{S}_u$
end for
return $\mathcal{S}_u$

Each identified subpattern will later be used to construct a branch in the DTMC for the corresponding user, as illustrated in Figure 5.2. By decomposing the pattern in this way, we ensure that the Markov chain model accurately captures the age dynamics for each user under the scheduling pattern.

5.2 DTMC Construction

Given a scheduling pattern, we aim to construct a DTMC that models the age evolution of each user over time. This Markov model is later used to find moments of AoI distribution for each user.

At the core of this construction is the notion of subpatterns, introduced in

the previous section. Each subpattern corresponds to a segment of the cycle between two consecutive update opportunities for a given user. These segments determine how the user's age evolves during the waiting period until the next scheduled update.

We focus on the geometric service model, where a user's transmission succeeds independently in each slot with a fixed probability q_n. Let a subpattern consist of the user sequence:

$$[p_1, p_2, \ldots, p_l]$$

where p_j is the user scheduled at position j. Each p_j defines a Markov state, and the transitions are defined as:

$$\text{State } j \to \begin{cases} j & \text{with probability } 1 - q_{p_j}, \\ j+1 & \text{with probability } q_{p_j}. \end{cases}$$

Each subpattern contributes a linear chain of transient states to the DTMC, terminating at a shared absorbing state that represents a successful transmission. These chains are combined into a global transient matrix B, and transitions to the absorbing state are collected in vector b. The resulting transition matrix $T \in \mathbb{R}^{(M+1)\times(M+1)}$ takes the following block structure:

$$T = \begin{pmatrix} B & b \\ \mathbf{0} & 1 \end{pmatrix} \tag{5.3}$$

where:

- $B \in \mathbb{R}^{M\times M}$ represents transitions among transient states,
- $b \in \mathbb{R}^{M}$ represents transitions to the absorbing state,
- Last row $[\mathbf{0} \quad 1]$ represents the self-loop at the absorbing state.

Example (continued). Consider the cyclic pattern $P = [1, 2, 3, 1, 2]$, and focus

on user $i = 1$, which appears twice. The corresponding subpatterns are:

$$\text{Subpattern}_1 = [1\ 2\ 3\ 1], \quad \text{Subpattern}_2 = [1\ 2\ 1]$$

Each subpattern generates a transient segment of the DTMC, determined by the success probabilities q_1, q_2, q_3. The resulting transition matrix is:

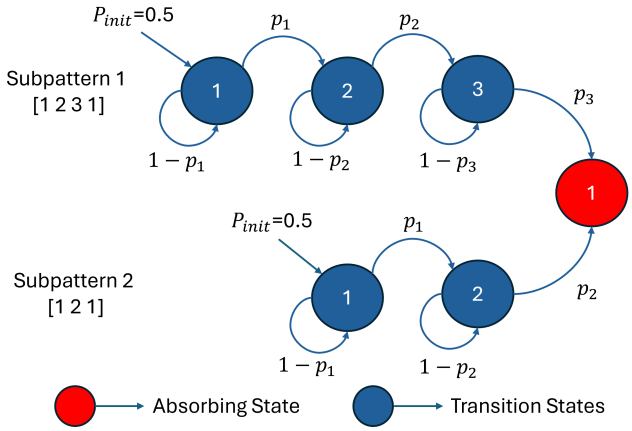

Figure 5.2: Markov chain with two subpatterns and a shared absorbing state.

$$T = \left[\begin{array}{ccccccc|c} 1-q_1 & q_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-q_2 & q_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-q_3 & q_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-q_1 & 0 & 0 & 0 & q_1 \\ 0 & 0 & 0 & 0 & 1-q_1 & q_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-q_2 & q_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1-q_1 & q_1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{5.4}$$

This completes the DTMC construction phase. In the next subsection 5.2.1, we describe how to compute the age-based probability distributions and VoI costs using this transition matrix. PMF computation is not a necessary process for calculating the system's mean cost; however, it is included in this section as it is used to validate the accuracy of the Markov-based model against simulation results. Additionally, the PMF is essential for evaluating Peak Value-of-Information (PVoI), as explained in the corresponding section on mean PVoI computation.

Following the PMF computation, we proceed with the mean system cost calculation using the moments of the AoI distribution, as explained in the corresponding chapter.

5.2.1 PMF Computation

To evaluate the VoI-based performance metrics for each user, we derive two key probability mass functions (PMFs):

- The PMF of the instantaneous age Δ_i at a randomly selected time slot:

$$\mathbb{P}(\Delta_i = k) = \alpha_i B_i^k h_i \tag{5.5}$$

- The PMF of the peak age Ψ_i, which represents the maximum age immediately before a successful transmission:

$$\mathbb{P}(\Psi_i = k) = \alpha_i B_i^{k-1} b_i, \quad k \geq 1 \tag{5.6}$$

These probability mass functions (PMF) are stated without full derivations, as they follow directly from the analytical framework presented in [42], which considers a similar context. The involved components are:

- **Transition matrix** $B \in \mathbb{R}^{M \times M}$**:** Represents all transitions among the transient states. It is assembled by combining the Markov chains derived from each subpattern.
- **Initial distribution** $\alpha \in \mathbb{R}^{M}$**:** Represents the probability of starting in each transient state. For users with m_i subpatterns, this is typically uniform across subpattern root states, i.e., each root state has weight $1/m_i$, and all other states are assigned zero.
- **Age overlap vector** $\hat{h} \in \mathbb{R}^{M}$**:** Specifies the states that overlap with age. Root states receive zero reward, and all other transient states are assigned

unit reward:

$$\hat{h}_j = \begin{cases} 0, & \text{if state } j \text{ is a root} \\ 1, & \text{otherwise} \end{cases}$$

- **Normalized age overlap vector $h \in \mathbb{R}^M$:** Defined to ensure that the computed age distribution is a valid PMF. It is calculated as:

$$h = \frac{\hat{h}}{\alpha(I - B)^{-1}\hat{h}} \tag{5.7}$$

- **Absorption vector $b \in \mathbb{R}^M$:** Contains the probabilities of transitioning from each transient state directly into the absorbing state. Nonzero entries appear only at the final states of subpatterns, where absorption is allowed.

Example (continued). Consider the user with two subpatterns extracted from the cyclic pattern $P = [1\ 2\ 3\ 1\ 2]$. The corresponding DTMC has 7 transient states and 1 absorbing state.

The user starts from either subpattern root with equal probability:

$$\alpha = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \end{bmatrix} \tag{5.8}$$

The unnormalized reward vector is:

$$\hat{h} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}^\intercal \tag{5.9}$$

The normalized version becomes:

$$h = \frac{\hat{h}}{\alpha(I - B)^{-1}\hat{h}} \tag{5.10}$$

Using this, we compute:

- The PMF of age via Equation (5.5),
- The PMF of peak age via Equation (5.6).

Once the age distribution $\mathbb{P}(\Delta_i = k)$ is computed for user i, we can evaluate the distribution of the VoI by applying the user's nonlinear cost function $f_i(k)$. This transformation reflects application-specific urgency and penalizes large age values more severely if needed.

Let the VoI function be defined as:

$$f_i(k) = a_i \Delta_i^2 + b_i \Delta_i + c_i \tag{5.11}$$

which maps each integer age k to a corresponding cost value $v = f_i(k)$. The goal is to construct a new probability mass function over cost values, denoted $\mathbb{P}(V_i = v)$, by aggregating the probabilities of age values k that map to the same cost:

$$\mathbb{P}(V_i = v) = \sum_{k \in \mathbb{N} \,:\, f_i(k) = v} \mathbb{P}(\Delta_i = k) \tag{5.12}$$

In other words, the probability that the VoI takes on value v is obtained by summing the probabilities of all age values k for which the cost function evaluates to v. This is a discrete re-indexing of the AoI-PMF into the VoI domain.

In addition to the average AoI, our model also supports the evaluation of Peak Value-of-Information (PVoI), which corresponds to the cost incurred at the peak age just before a successful update. This is computed using the peak age distribution $\mathbb{P}(\Psi_i = k)$, where Ψ_i denotes the number of time slots the system spends in transient states before absorption. Once this PMF is obtained, the same re-indexing method applies:

$$\mathbb{P}(\Psi_i = v) = \sum_{k \in \mathbb{N} \,:\, f_i(k) = v} \mathbb{P}(\Psi_i = k) \tag{5.13}$$

This enables analysis of not only average behavior but also worst-case costs experienced immediately before each information update.

To validate this transformation, we compare the theoretical age-based cost distributions obtained via our absorbing Markov chain model with empirical distributions derived from Monte Carlo simulations.

We consider a cyclic scheduling pattern given by:

$$P = [1,\ 2,\ 3,\ 1,\ 2,\ 2,\ 2,\ 2,\ 1]$$

where each entry denotes the scheduled user at each decision instant. The service times are geometrically distributed with success probabilities:

$$q = [0.7,\ 0.5,\ 0.2]$$

for users 1, 2, and 3, respectively. Here, each q_n represents the probability that a transmission attempt by user n successfully completes in a given slot. We assume equal user weights, and define the cost function as $f(\Delta) = \Delta^2$. Our focus is on user 1 to evaluate the VoI and PVoI distributions.

As shown in Figures 5.3 and 5.4, the analytical results closely match simulation outputs, demonstrating the validity of our Markov chain-based analysis.

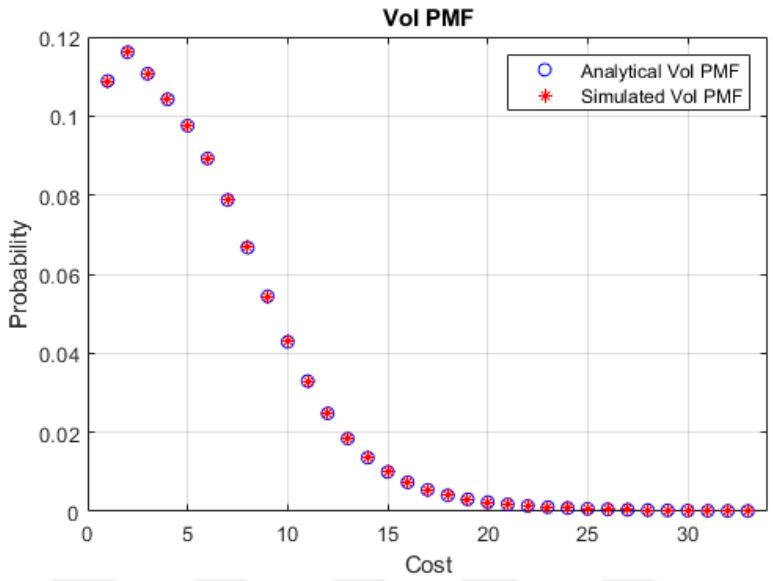

Figure 5.3: Comparison of analytical and simulated PMFs for VoI under cost mapping.

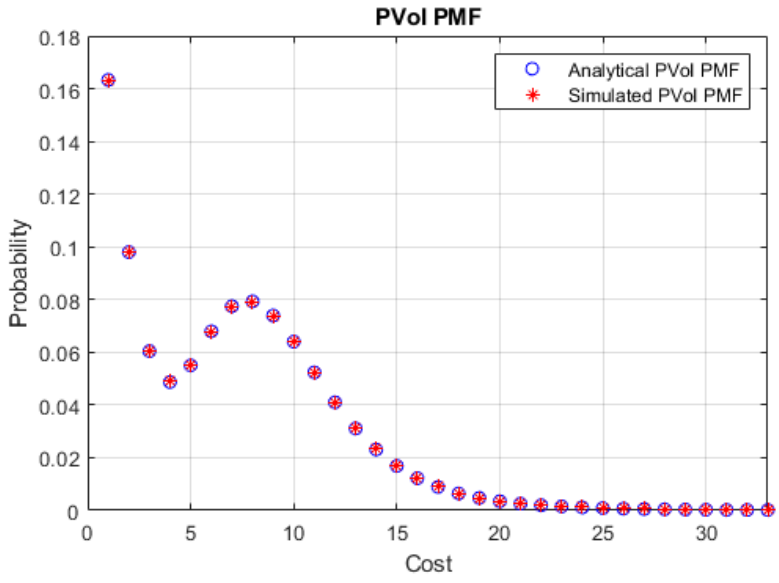

Figure 5.4: Comparison of analytical and simulated PMFs for Peak VoI under cost mapping.

5.3 Expectation Calculation

Once the transition matrix T is constructed, along with the associated transient block B, the initial distribution vector α, and the reward vector $\hat{h}$, we can compute the expected VoI using matrix-based formulations. This section details how to evaluate both the average and peak cost metrics using the discrete-time absorbing Markov chain model.

5.3.1 Age Distribution and Moments

Let $N_i = (I - B_i)^{-1}$ denote the fundamental matrix of the Markov chain, where I is the identity matrix. This matrix captures the expected number of visits to each transient state before absorption.

To obtain a valid PMF over the age states, the unnormalized reward vector $\hat{h} \in \mathbb{R}^M$ is normalized using:

$$h = \frac{\hat{h}}{\alpha N \hat{h}}, \tag{5.14}$$

which ensures that the resulting PMF sums to one.

Using this normalization, the PMF of the instantaneous AoI Δ_i is given in closed form as:

$$\mathbb{P}(\Delta_i = k) = \alpha_i B_i^k h_i, \quad k \geq 0. \tag{5.15}$$

Accordingly, the first and second moments of the age distribution are computed via:

$$\mathbb{E}[\Delta_i] = \sum_{k=0}^{\infty} k \cdot \mathbb{P}(\Delta_i = k) = \alpha_i B_i N_i^2 h_i, \tag{5.16}$$

$$\mathbb{E}[\Delta_i^2] = \sum_{k=0}^{\infty} k^2 \cdot \mathbb{P}(\Delta_i = k) = \alpha_i B_i (I + B_i) N_i^3 h_i. \tag{5.17}$$

5.3.2 Expected Value-of-Information

Given the user-specific cost function in Equation (5.11), the expected VoI cost can be expressed in two equivalent closed forms. Using the moments of Δ_i, the expected cost is evaluated using (5.17) as:

$$\mathbb{E}[f_i(\Delta_i)] = a_i \cdot \mathbb{E}[\Delta_i^2] + b_i \cdot \mathbb{E}[\Delta_i] + c_i. \tag{5.18}$$

Expressions in (5.18) represent the closed-form evaluations of the expected VoI and can be used for analytical or computations.

5.3.3 Peak Age Distribution and Moments

Let $N_i = (I - B_i)^{-1}$ denote the fundamental matrix of the Markov chain for source i, where I is the identity matrix. This matrix captures the expected number of visits to each transient state before absorption.

Let $b \in \mathbb{R}^M$ be the vector of absorption probabilities, where each entry represents the probability of transitioning from a given transient state to the absorbing state. The probability that the peak AoI (i.e., the age at the moment of a successful update) for source i equals k is given in closed form as

$$\mathbb{P}(\Psi_i = k) \;=\; \alpha_i B_i^{k-1} b, \qquad k \geq 1, \tag{5.19}$$

where α_i is the initial distribution over the transient states. Using this PMF, the first and second moments of the peak-AoI distribution are

$$\mathbb{E}[\Psi_i] \;=\; \sum_{k=1}^{\infty} k\, \mathbb{P}(\Psi_i = k) \;=\; \alpha_i N_i^2 b, \tag{5.20}$$

$$\mathbb{E}[\Psi_i^2] \;=\; \sum_{k=1}^{\infty} k^2\, \mathbb{P}(\Psi_i = k) \;=\; \alpha_i (I + B_i) N_i^3 b. \tag{5.21}$$

These expressions give closed-form evaluations of the peak-age moments, leveraging the fundamental matrix N_i and the absorption vector b. Unlike time-average AoI, which measures ongoing staleness, peak AoI characterises the age observed immediately before each successful update. This metric is important for applications that are sensitive to worst-case information freshness.

5.3.4 Expected Peak Value-of-Information

Given the same user-specific cost function in (5.11), we can evaluate the *peak* VoI i.e., the cost incurred immediately before each successful update using the moments of the peak-AoI random variable Ψ_i. With the peak-AoI moments in (5.21), the expected peak cost evaluated as:

$$\mathbb{E}\big[f_i(\Psi_i)\big] \;=\; a_i \cdot \mathbb{E}[\Psi_i^2] \;+\; b_i \cdot \mathbb{E}[\Psi_i] \;+\; c_i. \tag{5.22}$$

Expression (5.22) thus provides a closed-form evaluation of the expected peak VoI, fully analogous to the time-average VoI in (5.18), but based on the peak moments.

5.4 DTMC Construction With Discrete Phase Type Service Distributions

This section introduces the construction of discrete-time Markov chains for modeling age evolution under general service time distributions. To capture arbitrary discrete distributions beyond geometric models, we adopt discrete phase-type (DPH) representations. The construction proceeds in two main steps: first, local DPH blocks are defined for each user's service process; then, these blocks are sequentially concatenated into a global absorbing Markov chain according to the cyclic scheduling pattern. This framework enables the observation of system

behavior under different service time distributions by embedding each user's dynamics into a unified Markovian structure. It provides the flexibility to model heterogeneous and non-memoryless service processes, and supports performance evaluation through both age and cost-based metrics within a common analytical framework.

5.4.1 Defining Local Phase-Type Blocks for User Service Times

A DPH distribution is defined by a pair (α, T), where:

- $T \in \mathbb{R}^{M \times M}$ is a substochastic matrix representing transitions among M transient phases,
- $\alpha \in \mathbb{R}^{1 \times M}$ is the initial distribution vector over these phases.

Absorption occurs when the service completes, and the probability of absorption at each phase is implicitly given by $\boldsymbol{r} = 1 - T\mathbf{1}$, where $\mathbf{1}$ is an all-ones column vector.

For each user $i \in \{1, 2, \ldots, N\}$, we assume a known discrete service time distribution over support $\{1, 2, \ldots, M_i\}$, represented by a probability vector:

$$\boldsymbol{p}^{(i)} = \left[p_1^{(i)}, p_2^{(i)}, \ldots, p_{M_i}^{(i)}\right] \tag{5.23}$$

where $p_k^{(i)} = \mathbb{P}[S_i = k]$ denotes the probability that user i's service is completed in exactly k time steps. This vector can represent any valid distribution: geometric, binomial, truncated Poisson, empirical histogram, etc.

To construct a DPH representation from this distribution, we define the following:

- The number of transient states is set to M_i, corresponding to the maximum

possible service duration.

- The initial distribution vector $\alpha^{(i)}$ is chosen as:

$$\alpha^{(i)} = [1 \quad 0 \quad 0 \quad \dots \quad 0] \in \mathbb{R}^{1 \times M_i}$$

indicating that service always begins in the first phase.

- The transition matrix $T^{(i)} \in \mathbb{R}^{M_i \times M_i}$ is constructed recursively using conditional probabilities. Specifically, for $k = 1, 2, \dots, M_i - 1$, the probability of transitioning from phase k to $k+1$ is given by:

$$T^{(i)}_{k,\,k+1} = 1 - \frac{p^{(i)}_k}{1 - \sum_{j=1}^{k-1} p^{(i)}_j} \tag{5.24}$$

while the probability of absorption from phase k is:

$$r^{(i)}_k = \frac{p^{(i)}_k}{1 - \sum_{j=1}^{k-1} p^{(i)}_j} \tag{5.25}$$

These ensure that the marginal completion probabilities match the target distribution $\boldsymbol{p}^{(i)}$. For the final phase $k = M_i$, we set full absorption: $T^{(i)}_{M_i,\,M_i} = 0$, $r^{(i)}_{M_i} = 1$.

The result is a Markov model $(\alpha^{(i)}, T^{(i)})$ that captures the probabilistic evolution of the service process for user i. These models will later be integrated into a global chain corresponding to the user's inter-update intervals under a fixed cyclic schedule, as detailed in the next construction step.

5.4.2 Concatenating DPH Blocks into the Global Transition Chain

Once the DPH models for each user's service distribution are constructed, the next step is to assemble these blocks into a global absorbing Markov chain that captures the age evolution during each inter-update interval. This is done by

sequentially concatenating the DPH blocks corresponding to each scheduled user in the subpatterns identified for the user of interest.

Let $\mathcal{S}_i = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_S\}$ denote the set of subpatterns for user i, where each subpattern $\mathcal{P}_s = [p_1^{(s)}, p_2^{(s)}, \ldots, p_{L_s}^{(s)}]$ defines the sequence of scheduled users between two consecutive transmissions of user i. For each user $p_j^{(s)}$ in the subpattern, the corresponding DPH block $(\alpha^{(p_j^{(s)})}, T^{(p_j^{(s)})})$ is inserted into the global transition matrix in sequence.

To form the full chain for user i, the following operations are performed:

1. The first DPH block in the first subpattern is placed at the top-left corner of the global transition matrix. Its initial distribution defines the starting distribution for the chain.
2. For each subsequent user $p_j^{(s)}$ in the subpattern (across all $s = 1, \ldots, S$), the DPH block $T^{(p_j^{(s)})}$ is appended below the current matrix, expanding the global state space. The absorbing state of the previous block is redirected to the entry state of the new block, effectively chaining them together.
3. This redirection is implemented by identifying the last row of the preceding DPH block (i.e., the absorbing transition) and modifying it to point to the initial state of the newly added block. The self-loop at the local absorbing state is removed to avoid premature termination.
4. For the final DPH block of the last subpattern, the absorbing transition is preserved and directed to a single global absorbing state at the end of the chain. This state represents the successful update for user i.
5. The initial distribution vector α of the global chain is constructed by aligning the starting vectors of each first DPH block in the subpatterns, and setting all other entries to zero.

The schematic structure of the resulting global transition matrix is illustrated below. Each block T_k corresponds to a user's DPH representation, and the arrows

represent redirected transitions between the absorbing phase of one block and the initial phase of the next:

$$T_{\text{global}} = \begin{bmatrix} \boxed{T_1} & & & & \\ \to & \boxed{T_2} & & & \\ & \to & \boxed{T_3} & & \\ & & \to & 0 & 1 \\ & & & 0 & 0 \end{bmatrix} \tag{5.26}$$

This procedure results in a block-stacked transition matrix with a single absorbing state, where each block corresponds to a scheduled user's service model. The age accumulates as the chain progresses through the blocks, and resets upon absorption. The structure represents the impact of scheduling order and service variability on the user's age evolution under the cyclic pattern.

Once the transition matrix is constructed, the analysis proceeds by computing the moments of the age distribution using the absorbing Markov chain model, followed by the evaluation of the VoI as described in Section 5.3.

To give an example, consider a user whose subpattern is [1 2 3], meaning that user 1 is followed by users 2 and 3 before its next scheduled transmission. The scheduled users exhibit distinct service time distributions modeled as:

$$\boldsymbol{p}^{(1)} = \left[\tfrac{1}{2},\ \tfrac{1}{2}\right] \tag{5.27}$$

$$\boldsymbol{p}^{(2)} = \left[\tfrac{4}{7},\ \tfrac{2}{7},\ \tfrac{1}{7}\right] \tag{5.28}$$

$$\boldsymbol{p}^{(3)} = \left[\tfrac{1}{8},\ \tfrac{3}{8},\ \tfrac{3}{8},\ \tfrac{1}{8}\right] \tag{5.29}$$

Each distribution is converted into a DPH representation using the conditional absorption formulation described in Section 5.4.1. The resulting DPH blocks have 2, 3, and 4 transient states respectively. These are connected in sequence according to the subpattern, by redirecting the absorbing transition of each block

to the entry state of the next. The final absorption leads to a global absorbing state. The full transition matrix $T_{\text{global}} \in \mathbb{R}^{12\times 12}$ for this construction is:

$$T_{\text{global}} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 3/7 & 0 & 4/7 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 2/3 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7/8 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4/7 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \cdots \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{5.30}$$

Here:

- Rows 1–2 correspond to user 1's service phases.
- Rows 3–5 correspond to user 2,
- Rows 6–9 represent user 3,
- Row 10 contains the final local absorption from user 3.
- Row 11 is the global absorbing state.
- Row 12 represents the initial reset point for the next update cycle.

Redirections (i.e., implicit transitions between blocks) are already incorporated numerically as nonzero entries connecting the absorbing state of one user's DPH block to the starting state of the next.

It is important to note that, although phase-type (PH) distributions offer a rich modeling framework capable of representing a wide range of distributions (including those with infinite support), our examples are given in discrete-time distributions with finite support.

Chapter 6

Optimization Framework for Scheduling Patterns

In this chapter, we present a scheduling pattern construction algorithm and compare it to a greedy baseline, both designed to minimize the AoI in a GAW system. Our primary objective is to explore a feasible method for building reasonably efficient scheduling patterns. We have developed a dedicated analytical function, introduced in Chapter 5, that calculates the mean AoI or the associated value cost for any given user scheduling pattern. Here, a pattern refers to a sequence indicating the scheduling order of users. For instance, in a system with three users, a pattern can be any sequence of length K consisting of user indices (e.g., [1,2,3,1,...]). Typically, the length K is greater than the number of users N, allowing users to appear multiple times within the pattern depending on their service characteristics and system priorities. The function accepts this scheduling pattern as input and returns the corresponding cost metric for the pattern. A scheduling pattern specifies the order in which users are served. In practical scenarios, user service demands can vary with a broad range of mean service times, variances, and possibly different priorities or weights. We define the scenario as homogeneous when all users have similar or identical service demands, while heterogeneity refers to differences among users' service requirements. If the system were entirely homogeneous, a Round Robin (RR) scheduling pattern would

naturally be optimal, and there would be no need for complex pattern searching algorithms. However, user service requirements may vary (heterogeneous scenarios) and this would cause the RR approach to be suboptimal for this case. Consequently, our objective is to develop pattern-building algorithms capable of handling diverse service demands, ensuring good performance across both homogeneous and heterogeneous scenarios. The algorithms presented in this chapter initialize the scheduling patterns with a Round Robin sequence, as it is generally considered a reasonable starting point in the absence of prior information about the system. Omitting any user from the initial pattern would result in an invalid schedule, violating basic scheduling requirements since every user must appear in the pattern at least once. Moreover, it is necessary to impose practical limits on the maximum length of the patterns considered, as excessively long patterns consume significant memory and computational resources, making them impractical for real-world deployment. Additionally, the computational complexity of evaluating the cost of patterns grows rapidly with increased pattern length, making brute-force iterative search approaches computationally prohibitive. Due to the substantial search space associated with pattern optimization, heuristic and metaheuristic algorithms offer practical solutions. These methods efficiently explore large search spaces and are particularly suitable for scenarios where exhaustive or brute-force methods become computationally infeasible.

In this context, we evaluate two distinct scheduling algorithms:

1. An existing method developed previously, known for producing high-quality results but suffering from computational inefficiency as the number of users grows [1].

2. A newly proposed Simulated Annealing (SA)-based metaheuristic, designed to balance performance and runtime efficiency. SA allows probabilistic exploration of the solution space and helps avoid premature convergence to local minima, making it capable of discovering high-quality scheduling patterns with lower mean computation time compared to the greedy method.

Through simulations and analysis, we aim to identify a scheduling approach

that has an effective trade-off between AoI reduction and computational feasibility, thereby enabling the construction of system cost minimizing patterns.

6.1 Simulated Annealing Based Pattern Optimization

Having defined the analytical framework for computing the expected age-based cost for a cyclic transmission pattern in the previous section, we now focus on the problem of finding a cost-effective pattern that minimizes the mean system cost. Due to the combinatorial nature of the pattern space, we employ a simulated annealing metaheuristic to search for good solutions efficiently. To contextualize the performance of the proposed simulated annealing approach, we also evaluate a baseline method based on insertion search, which has been previously used in the literature for constructing cyclic schedules [1]. In our numerical experiments, we compare the SA-based method against this baseline to assess its performance. Let the cost function for a given pattern P be defined as Equation (5.2), where $\mathbb{E}[f_i(\Delta_i)]$ is computed as described in the previous section using the absorbing Markov chain model. The goal is to find a cyclic pattern $P^\star$ such that:

$$P^\star = \arg\min_{P \in \mathcal{P}} J(P) \tag{6.1}$$

where $\mathcal{P}$ denotes the space of feasible cyclic patterns. We use the simulated annealing algorithm described below to explore the possible pattern space.

Simulated Annealing Algorithm (initPattern, N, weights)

$P_{\text{curr}} \leftarrow$ initPattern, $\quad J_{\text{curr}} \leftarrow \text{EvaluateCost}(P_{\text{curr}})$
$P_{\text{best}} \leftarrow P_{\text{curr}}, \quad J_{\text{best}} \leftarrow J_{\text{curr}}$
$T \leftarrow T_0$
for $k = 1$ to maxIter **do**
  $P_{\text{new}} \leftarrow \text{GenerateNeighbor}(P_{\text{curr}})$
  $J_{\text{new}} \leftarrow \text{EvaluateCost}(P_{\text{new}})$
  $\delta \leftarrow J_{\text{new}} - J_{\text{curr}}$
  if $\delta < 0$ **or** $\text{rand}() < \exp(-\delta/T)$ **then**
    $P_{\text{curr}} \leftarrow P_{\text{new}}, \quad J_{\text{curr}} \leftarrow J_{\text{new}}$
    if $J_{\text{curr}} < J_{\text{best}}$ **then**
      $P_{\text{best}} \leftarrow P_{\text{curr}}, \quad J_{\text{best}} \leftarrow J_{\text{curr}}$
    end if
  end if
  $T \leftarrow \gamma T$
end for
return $P_{\text{best}}, J_{\text{best}}$

At each iteration, a neighboring pattern P_{new} is generated from the current pattern P_{curr}, and its cost $J(P_{\text{new}})$ is evaluated. The new pattern is accepted with probability:

$$\mathbb{P}_{\text{accept}} = \begin{cases} 1, & \text{if } J(P_{\text{new}}) < J(P_{\text{curr}}), \\ \exp\left(-\dfrac{J(P_{\text{new}}) - J(P_{\text{curr}})}{T}\right), & \text{otherwise,} \end{cases} \tag{6.2}$$

where T is the temperature parameter, which gradually decays according to a cooling schedule $T \leftarrow \gamma T$ with $0 < \gamma < 1$.

6.1.1 Neighbor Generation

Given the current pattern P, a neighboring pattern P' is generated through a stochastic perturbation process. With probability $p_{\text{append}} = 0.3$, a randomly

selected user $u \in \{1, \ldots, N\}$ is appended to the end of the pattern. Alternatively, with probability $p_{\text{swap}} = 0.7$, two randomly chosen positions within the pattern are swapped. The values p_{append} and p_{swap} were determined empirically through trial and error to balance exploration and exploitation during the search process. This randomized mechanism facilitates both exploration of new regions in the search space through insertions and local refinement of the current solution via swaps.

6.1.2 Cost Evaluation

For each candidate pattern P, the cost is evaluated via the Markov chain-based method described in the previous section. This optimization method allows the system to adaptively discover efficient cyclic schedules that reflect both user-specific priorities and non-linear staleness penalties.

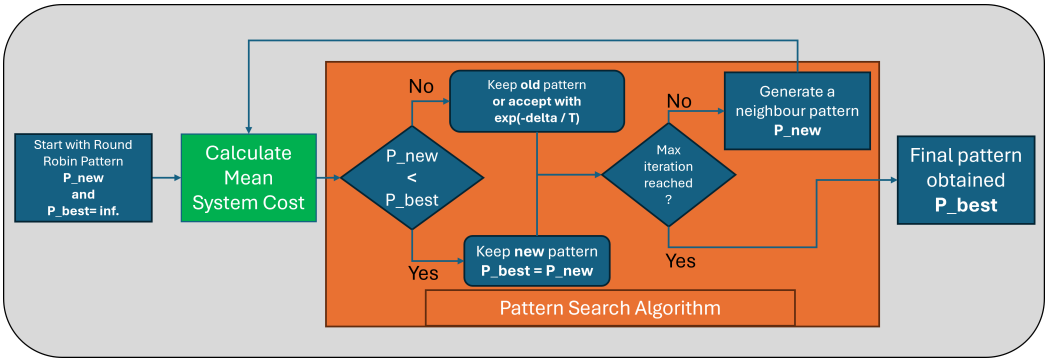

Figure 6.1: Flowchart of the pattern search algorithm for minimizing system-wide VoI

Figure 6.1 illustrates the overall flow of the pattern search algorithm. The process begins with a Round Robin pattern, which serves as an initial, balanced scheduling sequence. At each iteration, a new candidate pattern is generated and its cost is evaluated based on the mean system VoI. If the new pattern improves the objective, it replaces the current best; otherwise, the algorithm continues exploring the search space until a maximum iteration limit is reached. The final output is the pattern with the lowest observed cost. This method enables the

discovery of efficient schedules by iteratively refining pattern candidates based on their performance.

6.2 Insertion Search Algorithm

The Insertion Search (IS) algorithm is a greedy heuristic designed to minimize the system-wide AoI presented in [1]. It begins with a Round Robin pattern of length equal to the number of users N, ensuring that each user is scheduled exactly once. Starting from this base pattern, users are iteratively inserted into the pattern to reduce system AoI. The IS algorithm explores possible insertions to find local improvements in the total system cost.

At each iteration, the algorithm considers every user and evaluates the cost of inserting them at all possible positions in the current pattern. If any insertion yields a lower cost, the current pattern is updated. The process terminates when:

- No further improvement is observed by inserting any user at any position, or
- The pattern reaches a pre-defined maximum length $K_{\max}$.

The algorithm effectively increases the frequency of scheduling for important users those with high weights or low service times. This prioritization reduces their contribution to the system-wide cost.

Pseudocode:

Algorithm 2 Insertion Search (IS) Algorithm

Input:
$N \leftarrow$ number of users
$w_n \leftarrow$ weight of each user
`ServiceStats` $\leftarrow$ per-user service statistics (mean, variance)
$K_{\max} \leftarrow$ maximum allowed pattern length
Initialize:
$P \leftarrow [1, 2, \ldots, N]$ ▷ Initial Round Robin pattern
$C \leftarrow \text{Cost}(P)$ ▷ Initial system cost
`found` $\leftarrow$ true
while `found` = true and length$(P) < K_{\max}$ **do**
    `found` $\leftarrow$ false
    `bestPattern` $\leftarrow P$
    `bestCost` $\leftarrow C$
    for each user $i = 1$ to N **do**
        for each position $j = 0$ to length(P) **do**
            $P_{\text{cand}} \leftarrow$ insert user i at position j in P
            $C_{\text{cand}} \leftarrow \text{Cost}(P_{\text{cand}})$
            if $C_{\text{cand}} <$ `bestCost` **then**
                `bestPattern` $\leftarrow P_{\text{cand}}$
                `bestCost` $\leftarrow C_{\text{cand}}$
                `found` $\leftarrow$ true
            end if
        end for
    end for
    $P \leftarrow$ `bestPattern`
    $C \leftarrow$ `bestCost`
end while
return P

Note: The `Cost`(P) function is evaluated using the discrete-time Markov chain (DTMC)-based analytical model introduced in Chapter 5. Exhaustive exploration ensures insertions are tried across all valid positions.

Chapter 7

Numerical Results

In this section, we present simulation results to validate the effectiveness of our proposed scheduling framework. We focus on the GAW scenario, where we highlight the benefits of minimizing user-specific cost functions instead of the standard AoI metric. The performance of two scheduling strategies, IS and SA, is compared under various cost profiles to demonstrate the importance of tailored optimization in heterogeneous systems. In numerical examples, AoI corresponds to the linear cost function $f(\Delta) = \Delta$, whereas VoI refers to any general cost function of the form given in (3.6).

In the upcoming experiments, any scheduling strategy labeled as "AoI-optimized" refers to patterns that are constructed by minimizing the average AoI (i.e., $\mathbb{E}[\text{AoI}]$) using the Insertion Search method, and then evaluated under a non-linear cost function $f(\text{AoI})$. Thus, even though the optimization is performed over linear AoI, the final performance is assessed using the VoI framework.

7.1 Experiment 1

In this experiment, we demonstrate the advantages of directly minimizing the cost function, as opposed to first minimizing the age and subsequently evaluating

performance using the cost function. Additionally, we evaluate the performance of various scheduling heuristics in the minimization process of average VoI or AoI. Figure 7.1 presents a comparative analysis of SA, IS and RR pattern construction methods, where RR serves as a static, non-adaptive baseline. Experiment parameters are given in 7.1.

Table 7.1: System parameters for the experiment 1

Parameter	**Description**
User Weights	Equal for all users
Service Time Distribution	Geometric distribution
Service Time Parameter	$q_i = 1/i$ for user i, $\mathbb{E}[S_i] = i$
Cost Function	$f(\Delta) = 15\Delta^2$ (same for all users)

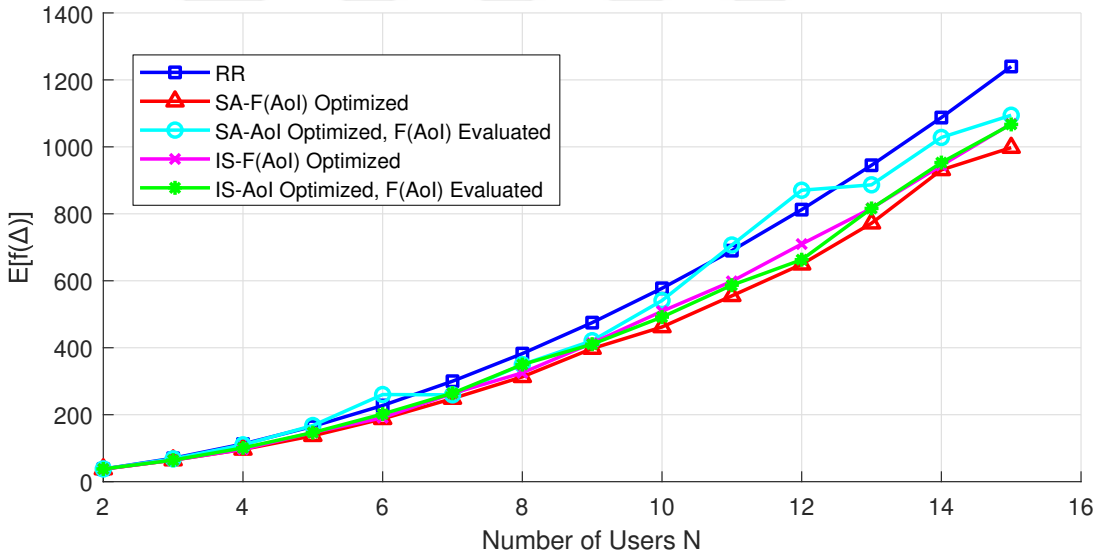

Figure 7.1: Comparison of SA and IS method; equal weight, user specific 1/N service time, cost is $f(\Delta) = 15\Delta^2$ for all users.

The results reveal several important trends. Most notably, the combination of SA with direct cost minimization consistently outperforms all other approaches in terms of average VoI. Second, although SA occasionally performs similar to IS at specific points-likely due to its randomized search trajectory it generally outperforms IS over the full range of user counts. This variation arises from the exploration exploitation balance inherent to SA, where worse solutions are accepted with controlled probability during early iterations, allowing the algorithm

to escape local optima.

Interestingly, when comparing the same optimization algorithm under different cost formulations, such as IS optimizing AoI directly versus minimizing the cost function the results are mixed. The two IS variants cross each other, showing that the algorithm's performance depends heavily on the choice of the objective function. This inconsistency can be attributed to IS's greedy behavior, which quickly converges to local minima and lacks the exploratory capacity needed to navigate more complex cost structures. In contrast, the SA based variants, particularly those optimizing the cost function directly, rarely intersect with their AoI optimized counterparts and consistently exhibit lower VoI values. This highlights the effectiveness of direct cost minimization when paired with SA, as the algorithm's inherent randomness helps it escape suboptimal regions and achieve significantly better outcomes in most scenarios. Overall, SA demonstrates greater robustness by maintaining exploratory behavior throughout its run, making it particularly suitable for non-convex objectives. Although SA does not guarantee global optimality and its performance can fluctuate due to stochastic elements, it generally yields lower average VoI, especially in scenarios involving non-linear costs and heterogeneous service models.

7.2 Experiment 2

In Figure 7.2, we use the experimental setting given in Table 7.2. Difference from Experiment 1 is the cost function changed to a linear form, also known as age, $f(\Delta) = \Delta$ in order to evaluate the performance of IS and SA algorithms.

Table 7.2: System parameters for the experiment 2

Parameter	**Description**
User Weights	Equal for all users
Service Time Distribution	Geometric distribution
Service Time Parameter	$q_i = 1/i$ for user i, $\mathbb{E}[S_i] = i$
Cost Function	$f(\Delta) = \Delta$ (same for all users)

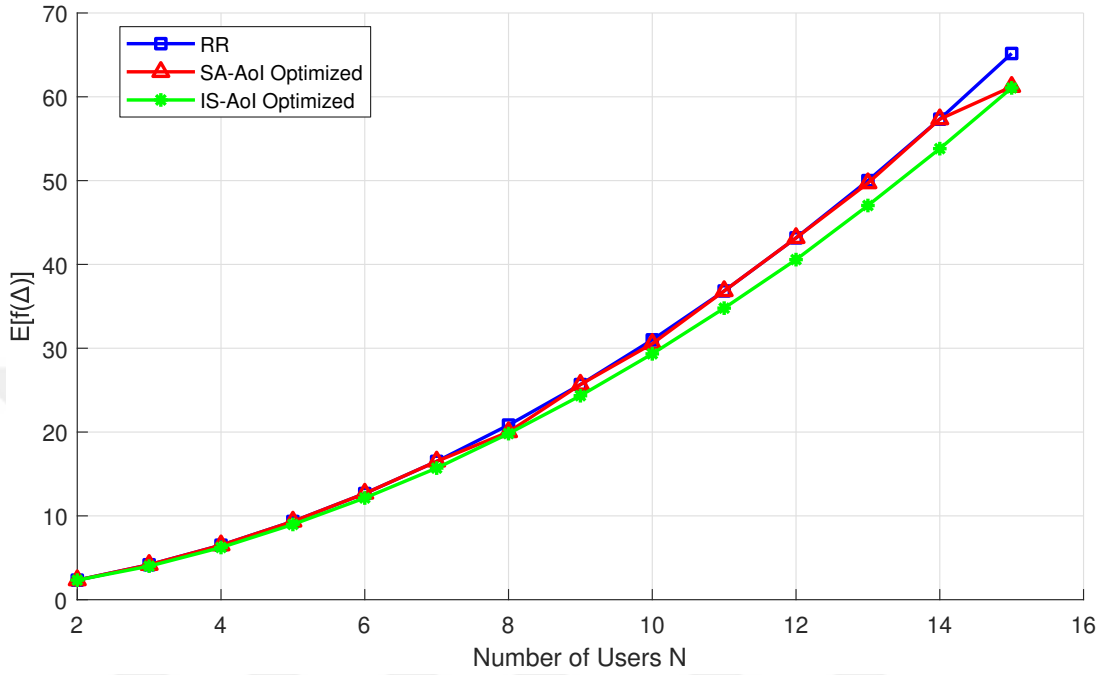

Figure 7.2: Comparison of SA and IS method; equal weight, user specific 1/N service time, simple age without user cost.

In this setting, we omit the results of cost evaluated patterns, as the focus is purely on minimizing raw AoI, without applying any nonlinear transformation. The results show that the performance of SA closely tracks that of the RR baseline occasionally performing slightly better or worse. At certain points, SA even marginally outperforms IS.

The effectiveness of IS in minimizing AoI has been established in prior work [1], and our findings are consistent with that conclusion: IS continues to outperform SA in this linear setting. We attribute this to the fact that greedy algorithms, such as IS, are particularly well suited for linear objective functions, where local search methods can efficiently converge to near optimal solutions.

7.3 Experiment 3

We repeated Experiments 1 and 2 using a heterogeneous weight model while keeping the service times equal across users, as detailed in Table 7.3.

Table 7.3: System parameters for the experiment 3

Parameter	**Description**
User Weights	$W_n = n^2$ for $n = 1, 2, \ldots, N$
Service Time Distribution	Geometric distribution
Service Time	Equal for all users; fixed as $S = 1$
Cost Function	For Figure 7.3 $f(\Delta) = 15\Delta^2$, For Figure 7.4 $f(\Delta) = \Delta$

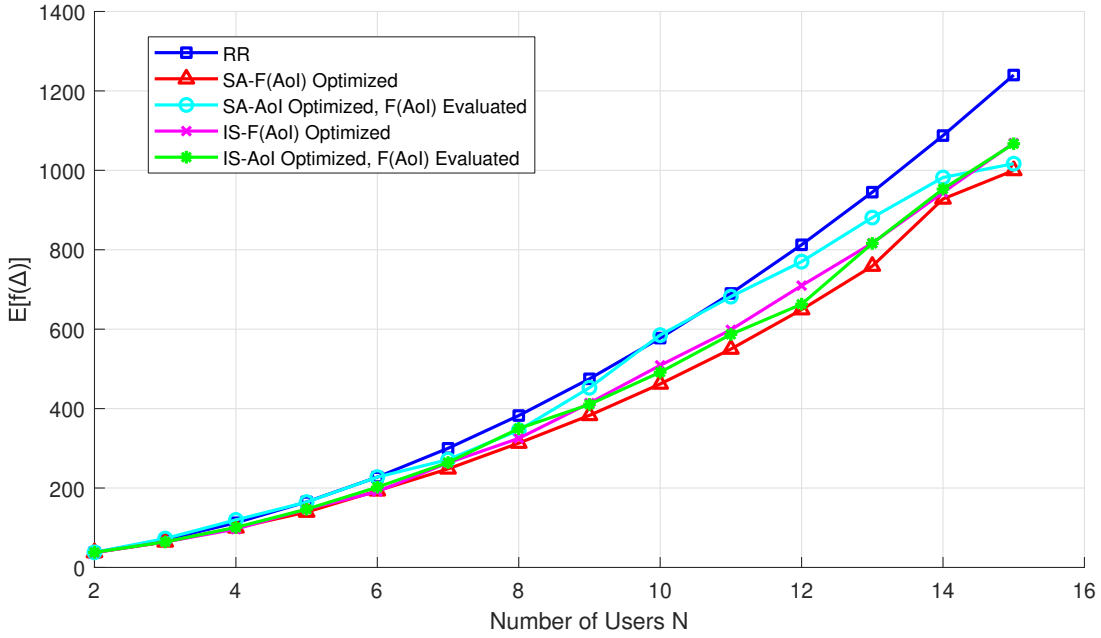

Figure 7.3: Expected cost $\mathbb{E}[f(\Delta)]$ versus number of users under the quadratic cost function $f(\Delta) = 15\Delta^2$, comparing IS and SA under different optimization targets, alongside the RR baseline.

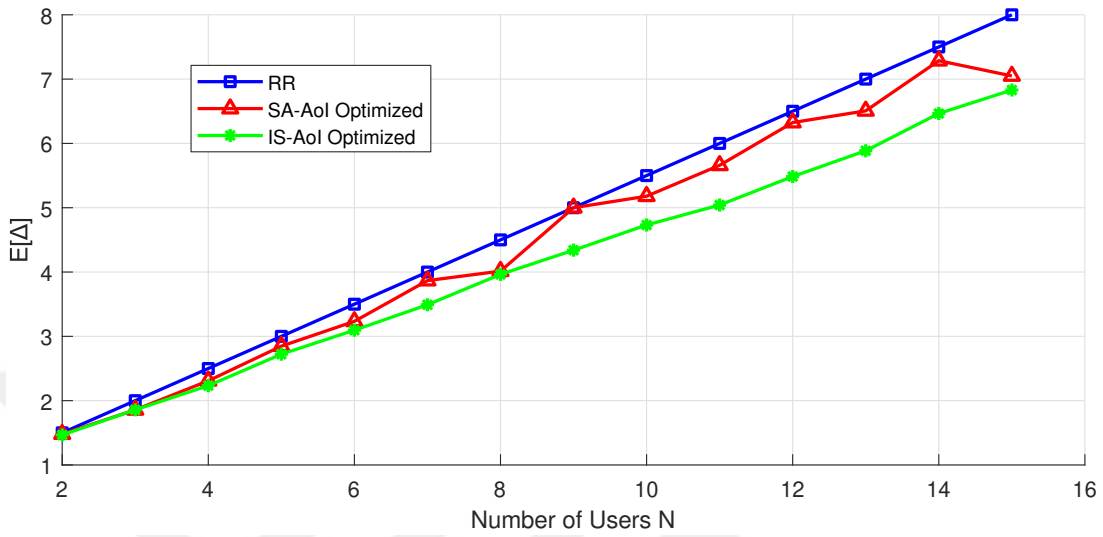

Figure 7.4: Expected cost $\mathbb{E}[f(\Delta)]$ versus number of users under the simple age function $f(\Delta) = \Delta$, comparing IS and SA under different optimization targets, alongside the RR baseline.

The results closely resemble those of the previous experiments. For the quadratic cost model, Simulated Annealing with direct cost minimization continues to yield the best performance. In contrast, for the linear cost case (i.e., the mean AoI domain) shown in Figure 7.4, the IS method still performs better, as explained in the discussion of earlier experiments (see Figure 7.3).

7.4 Experiment 4

In this experiment we consider the case where user-specific cost functions are introduced to capture heterogeneous urgency levels across users as illustrated in Figure 7.5. The parameters used in this system are presented in the Table 7.4.

Table 7.4: System parameters for the experiment 4

Parameter	**Description**
User Weights	$W_n = n^2$ for $n = 1, 2, \ldots, N$
Service Time Distribution	Geometric distribution
Service Time	Equal for all users; fixed as $S = 1$
Cost Function	$c_i(k) = \alpha_i \cdot k^2$, with $\alpha_i \sim \mathcal{U}(1, 10)$
Monte Carlo Count	20

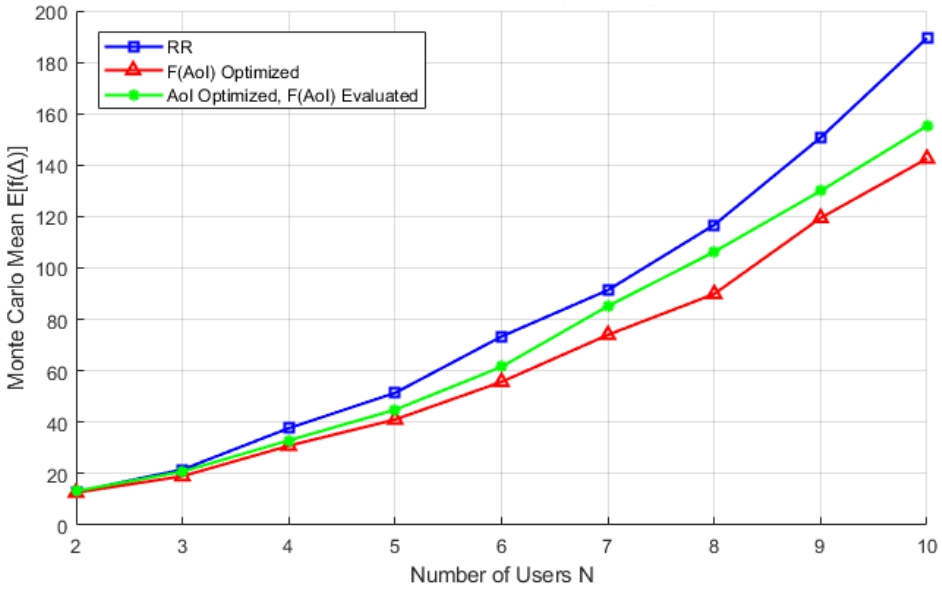

Figure 7.5: Effect of heterogeneous quadratic user specific cost functions.

As depicted in the figure, the performance gap between the strategies widens as cost heterogeneity increases. This result emphasizes the importance of cost-aware scheduling in systems with diverse user priorities and sensitivity to information staleness.

7.5 Experiment 5

In this experiment, we examine the pattern construction mechanisms of IS and SA by analyzing their minimization iterations and resulting pattern lengths in both the AoI and VoI domains. The analysis is conducted using the parameters

provided in Table 7.6 for an arbitrarily selected set of 7 users.

Table 7.5: System parameters for the experiment 5

Parameter	**Description**
User Weights	$W_n = n^2$ for $n = 1, 2, \ldots, N$
Service Time Distribution	Geometric distribution
Service Time	Equal for all users; fixed as $S = 1$
Cost Function	For Figure 7.6$f(\Delta) = \Delta$, For Figure 7.7$f(\Delta) = 15\Delta^2$,

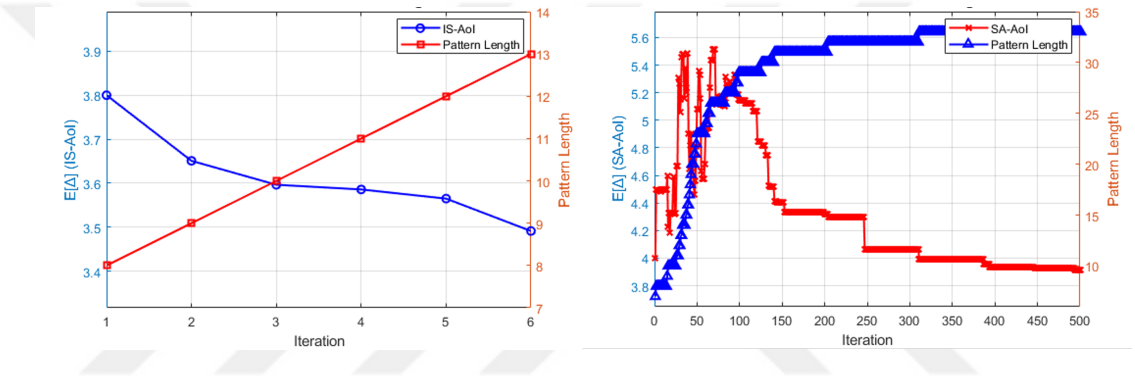

Figure 7.6: Pattern length and AoI cost evolution over iterations for IS (left) and SA (right) under linear cost function $f(\Delta) = \Delta$.

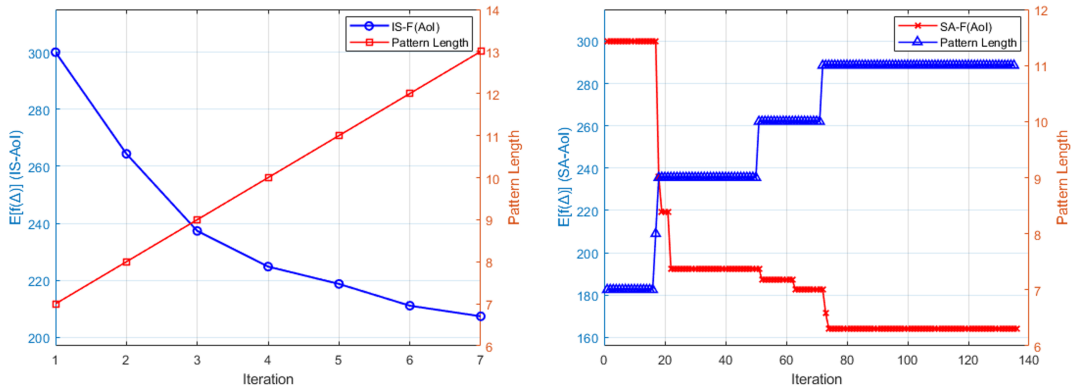

Figure 7.7: Pattern length and cost evolution for IS (left) and SA (right) under quadratic cost function $f(\Delta) = 15\Delta^2$.

When $f(\Delta) = \Delta$, large AoI values are penalized mildly, allowing occasional long gaps if offset by lower costs elsewhere. This leads to longer, more spread-out update cycles. In contrast, a quadratic cost like $f(\Delta) = 15\Delta^2$ penalizes high AoI

values more aggressively, thus pushing SA to avoid long inter-update intervals and favor tighter, shorter patterns.

The cost function also affects SA's acceptance probability, $\mathbb{P}_{\text{accept}} = \exp(-\Delta/T)$, where Δ is the cost difference. Under linear costs, Δ tends to be small, preserving exploration even at low temperatures. With quadratic costs, larger Δ values quickly suppress $\mathbb{P}_{\text{accept}}$, making the algorithm behave greedily and converge prematurely.

In summary, steeper cost functions like $15\Delta^2$ reduce both cycle length and exploration in SA, while smoother ones support more diverse schedules and better global search behavior.

7.6 Experiment 6

Table 7.6: System parameters for the experiment 6

Parameter	**Description**
User Weights	$W_n = n^2$ for $n = 1, 2, \ldots, N$
Service Time Distribution	Geometric distribution
Service Time	Equal for all users; fixed as $S = 1$
Cost Function	For Figure 7.8 $f(\Delta) = 15\Delta^2$

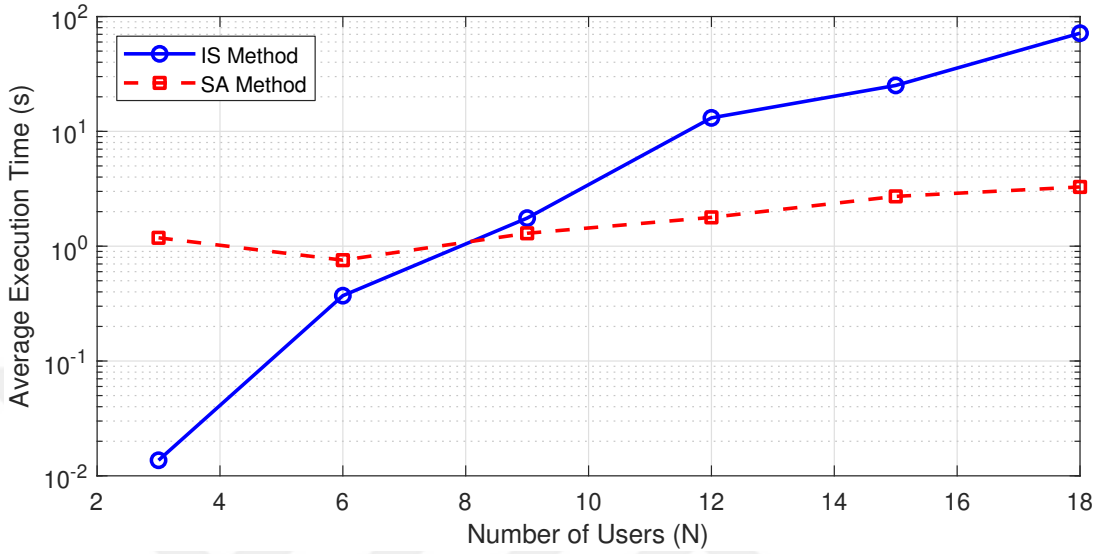

Figure 7.8: Average execution time (seconds) vs. number of users (log scale) for IS and SA

The Insertion Search (IS) algorithm builds the schedule step by step by testing all possible insertions. At each step, it considers up to $N \times K$ candidate patterns, where N is the number of users and K is the current cycle length. Since each candidate needs $\mathcal{O}(NK)$ operations to evaluate, the total per-iteration cost becomes $\mathcal{O}(N^2K^2)$. The process continues until the cycle reaches its optimal length K^*, so the overall complexity becomes $\mathcal{O}(N^2K^{*3})$, which can be quite slow for large values of N.

Simulated Annealing (SA), on the other hand, explores neighboring solutions in constant time and evaluates each one in $\mathcal{O}(N)$. With a fixed number of iterations K, the total runtime is $\mathcal{O}(\mathtt{K} \cdot N)$, making it more scalable for larger systems.

Figure 7.8 shows the runtime of both algorithms (on a log scale) for different user counts. For small N, IS is faster because the cycle length is short and the computation remains manageable. However, as N grows, the IS curve increases sharply due to the cubic growth in cost, while SA keeps a nearly linear trend. All runtimes were obtained with `MATLAB`'s `timeit` function which repeatedly executes the target routine until measurement variance stabilises, then

reports a robust average and were recorded on the author's personal laptop (Intel Core i7, 16 GB RAM).

7.7 Experiment 7

Unlike previous experiments where geometric service time distributions were used, this experiment aims to evaluate performance under alternative service time models. Specifically, we utilize DPH distributions to explore how the optimization performs under more general, non-memoryless conditions. The objective is to demonstrate that direct optimization of the cost function remains effective even when the service time model deviates from the geometric assumption. For this purpose, the system parameters summarized in Table 7.7 were used.

Table 7.7: System parameters for the experiment 7

Parameter	**Description**
User Weights	$W_n = n^2$ for $n = 1, 2, \ldots, N$
Service Time Distribution	Discrete; for each user, one of the following is randomly selected: – Uniform over {1,...,13} – Geometric with $p = \frac{1}{10}$ – Poisson with $\lambda = 5$, truncated to {0,...,12}
Service Time	User-dependent; sampled from the assigned distribution
Cost Function	For Figure 7.9, $f(\Delta) = 15\Delta^2$

In this experiment, we used the IS method for optimizing AoI, based on its superior performance in AoI minization as shown in earlier experiments compared to SA. Conversely, for direct cost function optimization, we applied the SA method due to its flexibility in navigating non-convex solution spaces. The results reveal that directly minimizing the cost function yields better performance than using an AoI-optimized pattern evaluated under a cost-based metric. These results confirm that our main claim remains valid under different service time distributions: directly optimizing the cost function yields lower system cost.

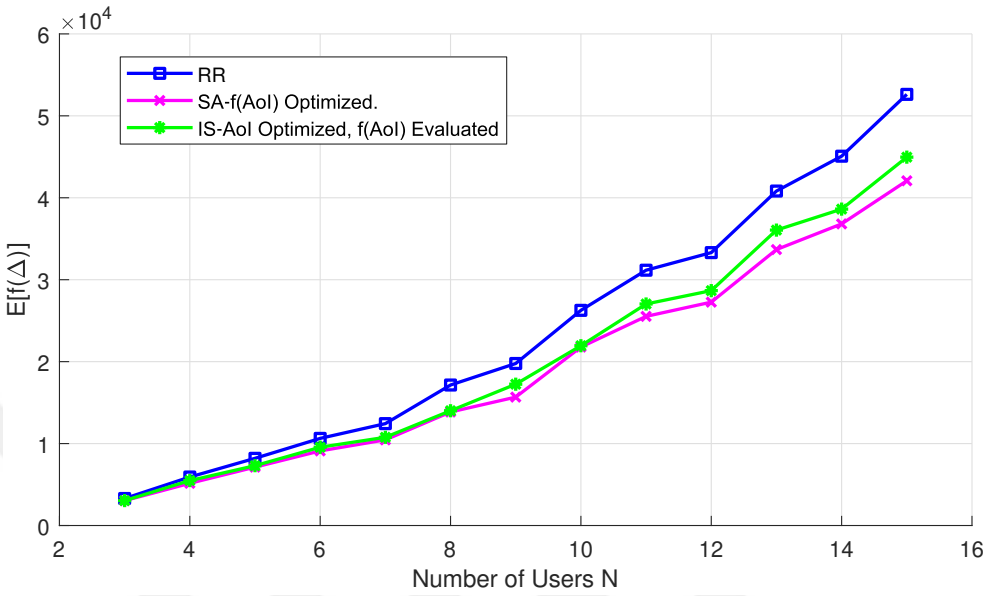

Figure 7.9: Performance of Direct Cost Optimization with Discrete Phase-Type Service Times

7.8 Experiment 8

Previously, in Figure 5.4 and Figure 5.3, we validated the accuracy of our analytical method based on absorbing Markov chains, showing that the simulation results closely align with the theoretical predictions. In this experiment, we further validate our absorbing Markov chain-based method by incorporating our results into a comparison based on the similar system model previously studied in [1]. The only difference between our work and the referenced study is that we consider a discrete-time setting, whereas the referenced work uses a continuous-time model. While we had previously verified the accuracy of the Markov-based approach through Monte Carlo simulations given in Figure 5.3, our aim here is to demonstrate its validity by directly comparing it against established methods from the literature. In the referenced work, a continous-time setting was considered, and two baseline approaches were evaluated: the Whittle index policy and the Insertion Search algorithm proposed by the authors for systems with n users.

Table 7.8: System parameters for the experiment 8

Parameter	**Description**
User Weights	$W_n = n^2$ for $n = 1, 2, \ldots, N$
Service Time Distribution	Geometric distribution
Service Time	Equal for all users; fixed as $S = 1$
Cost Function	Simple Age, $f(\Delta) = \Delta$

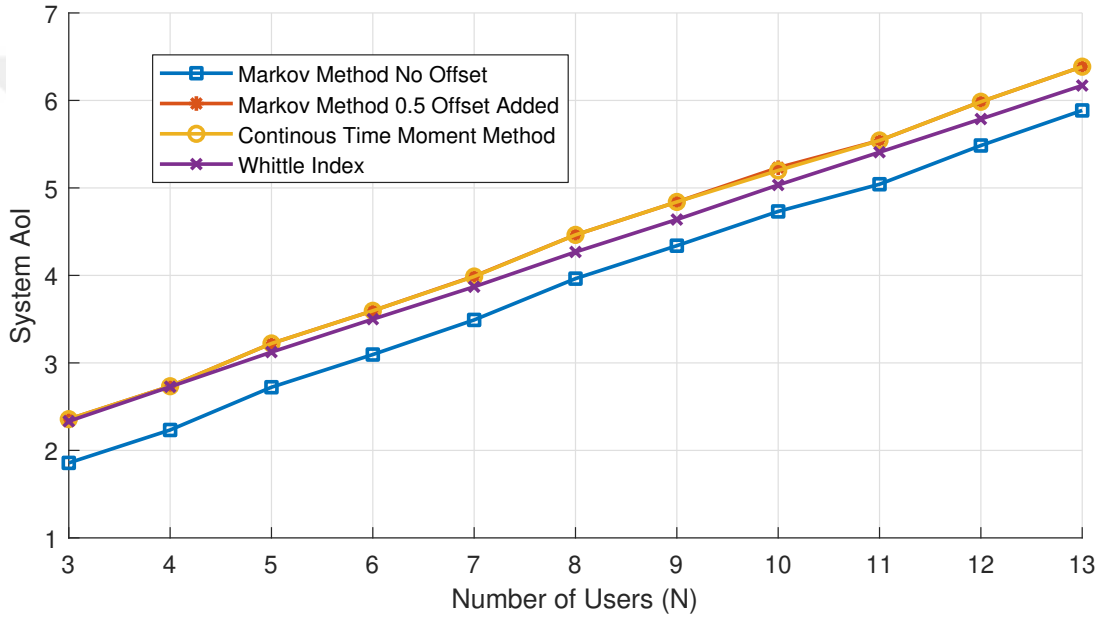

Figure 7.10: Comparison of AoI performance across methods for increasing number of users. Continuous-time and Whittle Index curves are re-created from [1]; other curves represent our method under discrete-time modeling with and without alignment offset

As shown in Figure 7.10, the curves labeled *Continuous Time Moment Method* and *Whittle Index* are reproduced from the reference study [1]. The curve labeled *Markov Method No Offset* represents the results of our proposed absorbing Markov chain-based method applied directly to the same system model. To align with the continuous-time results in [1], we also include an adjusted version (*Markov Method 0.5 Offset Added*) where a constant offset is applied to account for the structural difference between discrete and continuous time modeling. The figure clearly shows that, after applying the appropriate offset, our method aligns with the continuous-time results reported in the literature. This validates the

correctness of our approach under the same system assumptions. The origin of the observed 0.5 offset can be explained as follows: In the discrete-time Markov chain model, time progresses in fixed slots, and AoI increases in integer steps $(0, 1, 2, \dots)$. An update is considered to be received at the end of a slot, so the AoI resets discretely. In contrast, the continuous-time model treats time as a continuum, and AoI can reset at any point within a time interval. This leads to an additional expected AoI component corresponding to the average "residual age" within a service interval. For example, with deterministic service time $S = 1$, the continuous-time model yields an average residual age of:

$$\frac{\mathbb{E}[S^2]}{2\mathbb{E}[S]} = \frac{1}{2} = 0.5,$$

which does not appear in the discrete-time model. This explains the systematic offset observed between the two methods. In general, the offset is given by:

$$\text{Offset} = \frac{\mathbb{E}[S^2]}{2\mathbb{E}[S]},$$

and becomes relevant whenever continuous and discrete models are compared.

Chapter 8

Conclusion

In this thesis, we studied the problem of minimizing information staleness in discrete-time status update systems using a novel framework that extends the conventional age of information metric which amounts to the time average of the AoI process. Motivated by the limitations of average AoI in heterogeneous or mission-critical applications, we adopted the VoI perspective, which allows each user to be assigned a distinct, possibly nonlinear cost function that captures the urgency or utility of timely updates. In particular, polynomial cost functions are investigated.

Our work builds upon the cyclic scheduling model and extends it in several key directions. First, we considered a discrete-time setting and introduced a method to compute the expected cost for any given cyclic scheduling pattern using an absorbing Markov chain framework. This analytical approach enables exact performance evaluation under arbitrary discrete service time distributions, rather than relying on low-order moments. Second, we proposed a simulated annealing-based metaheuristic to efficiently explore the space of scheduling patterns, and demonstrated its superiority over greedy insertion methods, particularly when minimizing nonlinear cost functions. Numerical experiments confirmed that the optimization of VoI-based cost functions results in notable performance improvements compared to conventional AoI minimization.

Overall, this thesis contributes a flexible and analytically observable framework as an alternative to age-aware scheduling in slotted information-update systems. By bridging theoretical modeling with practical optimization techniques, it lays a foundation for future research in diverse real-world applications where update utility varies across users or contexts, such as industrial control, autonomous systems, and mission-critical monitoring scenarios.

Bibliography

[1] E. O. Gamgam, N. Akar, and S. Ulukus, "Cyclic scheduling for age of information minimization with generate at will status updates," in *2024 58th Annual Conference on Information Sciences and Systems (CISS)*, 2024, pp. 1–6.

[2] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2731–2735.

[3] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2011, pp. 350–358.

[4] C. Liu, Y. Qiu, M. Chen, and Y. Cai, "A vehicular network multipath management algorithm based on vehicular traffic," *Vehicular Communications*, 2025.

[5] S. Kaul, R. Yates, and M. Gruteser, "On piggybacking in vehicular networks," in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, 2011, pp. 1–5.

[6] 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*, vol. 57, no. 12, pp. 72–77, 2019.

[7] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, "Enhancements of v2x communication in support of cooperative autonomous driving," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 64–70, 2015.

[8] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Controlling the age of information: Buffer size, deadline, and packet replacement," in *MILCOM 2016-2016 IEEE Military Communications Conference*, 2016, pp. 301–306.

[9] K. R. Chevli, P. Y. Kim, A. A. Kagel, D. W. Moy, R. S. Pattay, R. A. Nichols, and A. D. Goldfinger, "Blue force tracking network modeling and simulation," in *MILCOM 2006-2006 IEEE Military Communications Conference*, 2006, pp. 1–7.

[10] J. Perry, R. Galliera, and N. Suri, "A machine learning approach to the determination of value of information to operators and applications on the tactical edge," *Procedia Computer Science*, vol. 205, pp. 137–146, 09 2022.

[11] M. James, Y. Sakae, and M. Hirohiko, "Value-of-information driven content presentation and filtering in military geographic information systems," in *Human Interface and the Management of Information. Visual Information and Knowledge Management*, 2019, pp. 545–554.

[12] T. M. Bojan, U. R. Kumar, and V. M. Bojan, "An internet of things based intelligent transportation system," in *2014 IEEE International Conference on Vehicular Electronics and Safety*, 2014, pp. 174–179.

[13] L. Li, "Application of the internet of thing in green agricultural products supply chain management," in *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, vol. 1, 2011, pp. 1022–1025.

[14] P. Popovski, Stefanović, J. J. Nielsen, E. de Carvalho, M. Angjelichinoski, K. F. Trillingsgaard, and A. S. Bana, "Wireless access in ultra-reliable low-latency communication (URLLC) ," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5783–5801, 2019.

[15] X. Zhang, K. Xiong, W. Chen, P. Fan, B. Ai, and K. B. Letaief, "Minimizing AoI in high-speed railway mobile networks: DQN-based methods," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[16] F. Alawad and F. A. Kraemer, "Value of information in wireless sensor network applications and the IoT: A review," *IEEE Sensors Journal*, vol. 22, no. 10, pp. 9228–9245, 2022.

[17] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Non-linear age of information in a discrete time queue: Stationary distribution and average performance analysis," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[18] A. Kosta, N. Pappas, A. Ephremides, and V. Angelkis, "The age of information in a discrete time queue: Stationary distribution and non-linear age mean analysis," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1352–1364, 2021.

[19] 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*, vol. 26, no. 6, pp. 2637–2650, 2018.

[20] 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*, vol. 20, no. 2, pp. 1263–1277, 2020.

[21] S. Kriouile, M. Assaad, and A. Maatouk, "On the global optimality of whittle's index policy for minimizing the age of information," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 572–600, 2021.

[22] C. Li, S. Li, Q. Liu, Y. T. Hou, W. Lou, and S. Kompella, "Eywa: A general approach for scheduler design in AoI optimization," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, 2023, pp. 1–9.

[23] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "The cost of delay in status updates and their value: Non-linear ageing," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4905–4918, 2020.

[24] A. Kosta, N. Pappas, V. Angelakis *et al.*, "Age of information: A new concept, metric, and tool," *Foundations and Trends® in Networking*, vol. 12, no. 3, pp. 162–259, 2017.

[25] R. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[26] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Transactions on Information Theory*, vol. 62, 06 2016.

[27] M. Moltafet, M. Leinonen, and M. Codreanu, "On the age of information in multi-source queueing models," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 5003–5017, 2020.

[28] 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*, vol. 26, no. 6, pp. 2637–2650, 2018.

[29] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 221–230.

[30] E. O. Gamgam and N. Akar, "Exact analytical model of age of information in multi-source status update systems with per-source queueing," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20 706–20 718, 2022.

[31] N. Akar and O. Dogan, "Discrete-time queueing model of age of information with multiple information sources," *IEEE Internet of Things Journal*, vol. 8, no. 19, pp. 14 531–14 542, 2021.

[32] Y.-P. Hsu, E. Modiano, and L. Duan, "Scheduling algorithms for minimizing age of information in wireless broadcast networks with random arrivals," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2903–2915, 2019.

[33] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1807–1827, 2019.

[34] 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*, vol. 28, no. 5, pp. 2215–2228, 2020.

[35] Y. Chen and A. Ephremides, "Minimizing age of incorrect information for unreliable channel with power constraint," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[36] J. Chen, J. Wang, C. Jiang, and J. Wang, "Age of incorrect information in semantic communications for noma aided xr applications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 5, pp. 1093–1105, 2023.

[37] C. Kam, S. Kompella, and A. Ephremides, "Age of incorrect information for remote estimation of a binary markov source," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 1–6.

[38] A. Nayak, A. E. Kalør, F. Chiariotti, and P. Popovski, "A decentralized policy for minimization of age of incorrect information in slotted aloha systems," in *ICC 2023-IEEE International Conference on Communications*, 2023, pp. 1688–1693.

[39] B. Chang, B. Kizilkaya, L. Li, G. Zhao, Z. Chen, and M. A. Imran, "Effective age of information in real-time wireless feedback control systems," *Science China Information Sciences*, vol. 64, pp. 1–14, 2021.

[40] R. Devassy, G. Durisi, G. C. Ferrante, O. Simeone, and E. Uysal-Biyikoglu, "Delay and peak-age violation probability in short-packet transmissions," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 2471–2475.

[41] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age and value of information: Non-linear age case," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 326–330.

[42] N. Akar and E. O. Gamgam, "Distribution of age of information in status update systems with heterogeneous information sources: An absorbing markov chain-based approach," *IEEE Communications Letters*, vol. 27, no. 8, pp. 2024–2028, 2023.

[43] N. Akar and O. Dogan, "Discrete-time queueing model of age of information with multiple information sources," *IEEE Internet of Things Journal*, vol. 8, no. 19, pp. 14 531–14 542, 2021.