Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Bachelor Thesis in Information Engineering

Optimal Timely Monitoring of Events over a Finite Time Horizon

Bachelor Candidate

Maksim Lukashkou

Student ID 2047104

Supervisor

Prof. Leonardo Badia

University of Padova

Academic Year
2023/2024

Abstract

Timely and effective monitoring of events is crucial across various domains, ranging from healthcare to finance and emergency responses. This thesis considers the problem of selecting optimal monitoring points for events occurring over a finite time horizon, where events follow diverse distributions. That is, the event has unknown timing but it is bounded within a finite time window and its probability density function is known. The primary objective is to minimize the age of information related to an event by apriori strategically placing monitoring points on a normalized time scale using optimization techniques that minimize a penalty function. The analysis considers single and multiple monitoring points. Moreover, several popular distributions for are considered the event. Closed-form solutions are derived when the statistical distribution of the event is known, while numerical simulations are employed for scenarios with multiple monitoring points or complex distributions. The findings reveal intriguing trends, including saturation and uniformity of monitoring times under certain distributions, demonstrated through graphical representations generated from numerical analyses.

Sommario

Il monitoraggio tempestivo ed efficace degli eventi è cruciale in vari settori, dalla finanza alla assistenza sanitaria e alle risposte di emergenza. Questa tesi considera il problema della selezione dei punti di monitoraggio ottimali per gli eventi che si verificano entro un orizzonte temporale finito, dove gli eventi seguono distribuzioni diverse. In altre parole, l'evento ha un momento sconosciuto ma è limitato entro una finestra temporale finita e la sua funzione di densità di probabilità è nota. L'obiettivo principale è quello di minimizzare l'età delle informazioni relative a un evento posizionando strategicamente i punti di monitoraggio su una scala temporale normalizzata utilizzando tecniche di ottimizzazione che minimizzano una funzione di penalità. L'analisi considera punti di monitoraggio singoli e multipli. Inoltre, vengono considerate diverse distribuzioni popolari per l'evento. Soluzioni in forma chiusa sono derivate quando la distribuzione statistica del evento è nota, mentre simulazioni numeriche sono impiegate per scenari con punti di monitoraggio multipli o distribuzioni complesse. I risultati rivelano tendenze intriganti, tra cui la saturazione e l'uniformità dei tempi di monitoraggio sotto alcune distribuzioni, dimostrate attraverso rappresentazioni grafiche generate da analisi numeriche.

Contents

1 Introduction

1.1 Overview of Age of Information

In many real-world applications, timely event monitoring is essential for effective decision-making and response. Whether in healthcare, finance, surveillance, transportation networks, or emergency services, the ability to quickly and accurately detect events can significantly impact outcomes. In these systems, the timeliness and freshness of information are crucial for performance and reliability. The concept of Age of Information (AoI) has emerged as a key metric for quantifying the freshness of information in systems.

1.1.1 Definition

Age of Information (AoI) is a metric used to quantify the freshness of information in a system that monitors events. It is defined as the time elapsed since the last received update was generated at the source. Mathematically, at any given time t the Age of Information $\Delta(t)$ can be expressed as a random process [27]:

$$\Delta(t) = t - u(t)$$

where $u(t)$ is the timestamp of the most recent update received before or at time t.

Several factors can affect the Age of Information (AoI) in real-time systems. One key factor is the timing of information retrieval, known as the "polling time"—the specific moment when we decide to check for updates. In this work, the objective is to strategically place monitoring points based on the event's distribution to minimize the AoI by reducing update delays. Thereby maximizing the freshness and relevance of the information available for decision-making processes. This finds applications in the following fields:

1.1.2 Applications

- **Real-Time Traffic Management**
  - Traffic management systems rely on real-time data from vehicles to monitor and control traffic flow. By optimizing polling delay, traffic control centers can ensure that they have the freshest information, enabling better decision-making for traffic light adjustments, congestion management, and route optimization. For example, during peak hours for certain intersection based on the historical data, the system increases the frequency of polling data from traffic sensors. This ensures that the information about traffic flow and congestion is updated more frequently. Thus, leading to a more efficient allocation of computing resources and energy.
- **E-Health**
  - Health monitoring systems, including wearable devices and remote patient monitoring solutions, rely on low Age of Information (AoI) to deliver accurate and timely health data. This is essential for facilitating prompt diagnosis and intervention. Many of these systems operate on small internal batteries, making it crucial to minimize the frequency of updates to extend battery life. This can be achieved by scheduling measurements during specific times of the day when the patient is more vulnerable to health issues. Such timing considerations can be informed not only by the clock but also by the patient's activity levels and other health indicators.
- **Trading**
  - Optimal monitoring is very important in the world of trading. For instance, during periods of high volatility, firms may increase the frequency of data polling from market feeds to capture fleeting price movements. This approach enables more effective use of algorithms and computational resources, ultimately enhancing trading performance.

- **Sensor Networks**
  - Sensor networks deployed for environmental monitoring, industrial automation, or smart cities are typically placed in the field with internal batteries and limited transmission access. Consequently, the frequency of data transmission is restricted. Therefore, it is essential to optimize the transmission of status updates to minimize Age of Information (AoI). This can be achieved by analyzing the distribution of the monitored events and identifying the most effective observation points.

- **Server Monitoring**
  - In the context of server monitoring, determining the optimal times to check the status of servers can greatly enhance the reliability and performance of IT infrastructure. By minimizing the AoI, administrators can ensure that they have the most recent data regarding server health, load, and performance. This approach can prevent potential downtime and quickly address issues, thereby maintaining high availability and efficiency in data centers and cloud environments.

- **Handling Interrupts**
  - Another crucial application of this optimization problem is in handling interrupts in computing systems. Determining when to poll devices for interrupts can optimize the balance between responsiveness and resource utilization. By strategically placing polling intervals, the system can minimize latency and ensure timely processing of device requests without unnecessary overhead. This is particularly important in real-time operating systems and embedded systems, where timely handling of interrupts is critical for system stability and performance.

1.1.3 Key Characteristics

It is important to note that, in addition to the update policy, several factors can influence Age of Information (AoI) in real-life scenarios. For example,

1. **Update Generation Rate**:
   - The rate at which new updates are generated can impact AoI. Higher update rates can potentially reduce AoI, provided the system can handle the increased traffic without introducing significant delays [17].

2. **Transmission Delay**:
    - The time it takes for an update to travel from the source to the destination affects AoI. Lower transmission delays lead to fresher information [26].

3. **Queueing Delay**:
    - Updates often need to wait in a queue before being transmitted, especially in systems with limited bandwidth or high traffic. Longer queueing times increase AoI [18].

4. **Update Loss**:
    - Packet losses or dropped updates due to network issues or buffer overflows can lead to increased AoI, as the receiver will rely on older information until the next successful update [23].

5. **Processing Delay**:
    - The time required to process an update once it reaches its destination also affects AoI. Faster processing reduces AoI [16].

6. **Update Policy**:
    - The strategy for sending updates (e.g., periodic updates, event-driven updates) can impact AoI. Optimal policies aim to minimize AoI by balancing the update frequency and system load [12].

7. **Network Congestion**:
    - High levels of network congestion can increase both transmission and queueing delays, leading to higher AoI [4].

8. **Resource Allocation**:
    - Allocation of resources such as bandwidth and computing power can affect the timeliness of updates. Adequate resources help maintain lower AoI [19].

9. **Network Topology**:
    - The structure and layout of the network can impact AoI. More complex topologies might introduce additional delays due to routing and multiple hops [10].

10. **Prioritization Mechanisms**:

    - Systems that prioritize newer updates or critical information can help reduce AoI by ensuring timely delivery of the most relevant data [9].

11. **Feedback Mechanisms**:

    - Mechanisms that provide feedback on the status of updates (e.g., acknowledgments) can help in managing and reducing AoI by ensuring updates are successfully received and processed [3].

12. **Synchronization**:

    - In systems where multiple sources provide updates, synchronization mechanisms can impact AoI by ensuring coherent and timely updates [20].

1.2 Related work

A lot of interesting work utilizes Age of Information as a primary metric. For example, [31] looks for optimal scheduling of sensor updates in real-time systems to minimize the age of information, considering the impact of transmission delays. It derives a closed-form expression for the average age of information and evaluates performance under different conditions. Another interesting application is shown in [14] that analyzes a status updating system where timely information at the destination is critical, using the AoI metric to assess freshness over an erasure channel. It determines the optimal failure tolerance for various erasure probabilities and provides bounds for average AoI and peak AoI, focusing on whether to continue sending the current update or switch to a new one based on the percentage of successful receptions.

More practical results include [29] that introduces the Age of Task-oriented Information (AoTI) metric to accurately measure system freshness in industrial wireless sensor networks, which rely on multiple types of sensing data for tasks like fire alarms. It aims to minimize long-term AoTI by optimizing access selections and sampling frequencies through a Learning-based Access selection and Sampling frequency Control (LASC) algorithm. Or [1] that proposes a Mutable Sensor Data Analytics approach to minimize Age of Information related metrics in the Internet of Underwater Things. It utilizes block-chain and gradient descendent learning to classify noise and irrelevant data in order to avoid transmitting them. Thus, reducing age of information for relevant data. Another interesting work [5] explores optimizing sensor data transmission schedules in smart agriculture and industry, where external sources provide non-controllable updates. By proposing an online dynamic programming method to minimize Age of Information (AoI), the study identifies optimal transmission intervals and addresses data redundancy challenges.

There are also many works that approach Age of Information optimization using game theory. [7] studies the problem where users participate in the data aggregation process aiming to maximize their individual utility, balancing global AoI and personal costs. This is useful to optimize modern federated learning algorithms. The paper that includes the same authors [6] also tackles the problem of optimal update policy. It aims to offer a new standpoint for timely status update considering both the AoI and the Age of Incorrect Information (AoII) for machine learning applications. Another work [13] uses federated learning to minimize network energy consumption for industrial IoT while managing AoI constraints.

The study [30] investigates how sensor correlation affects the Age of Information (AoI) under concurrent and time-division multiple access scenarios, showing that concurrent transmissions improve AoI more significantly, and demonstrates through simulations that ML techniques can optimize transmission policies and reduce resource usage without impacting AoI.

It is worth noting that techniques developed to optimize Age of Information can also be used in other settings. Work [2] argues that both energy provision and data networking issues stem from efficient resource management, which can be addressed using dynamic programming, and demonstrates how energy management in smart micro grids can be optimized using a framework designed for the optimization of age of information.

However, Age of Information cannot be used for certain scenarios. For example, [15] mentions that AoI cannot address non-linearity that can be useful in designing novel sensing approaches. Therefore, the authors employed the urgency of information (UoI) framework for their study by combining the timeliness and context associated with information updates.

Another interesting metric is Value of Information (VoI) used in the study [11]. It is used to account for remote process estimation and timing constraints.

This metric allows the authors to study the problem of the pragmatic communication with multiple clients.

Next chapters will discuss the mathematical formulation of the problem, the derivation of the objective function, and the optimization techniques used to find the optimal monitoring points.

2
Uniformly Distributed Event With Two Monitoring Points

This chapter addresses the problem of finding optimal monitoring points for events that are uniformly distributed over a finite time horizon. Specifically, we focus on the scenario where two monitoring points are to be optimally placed to minimize the Age of Information (AoI). In the case of one monitoring point, results for the distribution with triangular shape were presented in [28].

2.1 Problem Statement

Given a time horizon $[0, 1]$, where an event can occur at any time $t \in [0, 1]$ with uniform probability, our goal is to determine the optimal placement of two monitoring points, t_1 and t_2, such that the AoI is minimized.

To formally state the problem:

- **Objective:** Minimize the expected Age of Information (AoI) for an event that follows uniform distribution over a normalized time-window $[0, 1]$.
- **Variables:** The locations of the two monitoring points, t_1 and t_2, where $0 \leq t_1 < t_2 \leq 1$.
- **Constraints:** The monitoring points must be placed within the time horizon $[0, 1]$.

By strategically placing the monitoring points, we aim to derive closed-form solutions or numerical approximations that yield the minimum AoI. This problem is significant as it lays the groundwork for more complex scenarios involving multiple events, more monitoring points, and varying distributions.

The remainder of this chapter will discuss the derivation of the objective function, and the optimization techniques used to find the optimal monitoring points.

2.2 Derivation

Suppose we have two monitoring points, t_1, t_2, such that $t_2 > t_1$ and we use a linear penalty function. The following figure shows how the penalty function changes depending on the relative positions of an event and a monitoring time on the time axis.

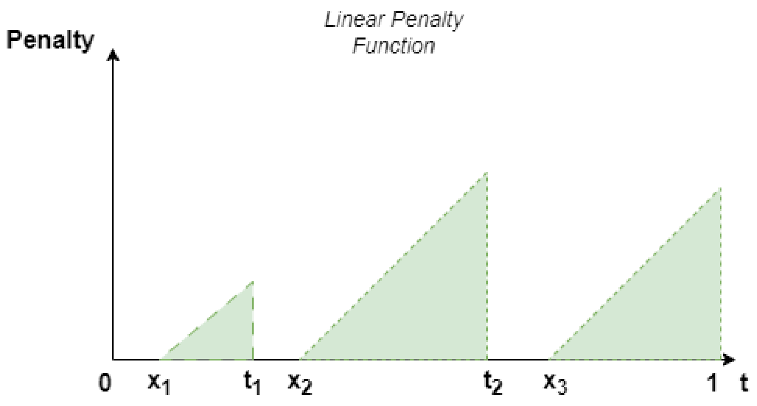

Figure 2.1: Linear penalty function

As shown in the figure, the penalty increases when the monitoring point is further away. When the monitoring point is before the event, the penalty continues to accumulate until the end of the time window.

By taking similar approach as in [28], we are able to derive the expected value of the total penalty.

Firstly, detection time for two monitoring points can be expressed as:

$$D(t_1, t_2, x) = t_1 + (t_2 - t_1)u(x - t_1) + (1 - t_2)u(x - t_2) \tag{2.1}$$

Then, the penalty function is the following:

$$p_{t,x}(u) = \begin{cases} u - x, & \text{if } x < u < D(t_1, t_2, x) \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

It equals to zero before the event has happened and after the event has been detected. Whereas it grows linearly after the event has happened but it is not yet detected.

Thus, the total penalty over a finite window for an event that occurs at time x given t_1, t_2 is an integral of the penalty over the time window:

$$P(t_1, t_2, x) = \int_0^1 p_{t,x}(u)du = \begin{cases} \int_x^{t_1} (u - x)\, du & \text{if } t_1 > x \\ \int_x^{t_2} (u - x)\, du & \text{if } t_2 > x > t_1 \\ \int_x^{1} (u - x)\, du & \text{if } 1 > x > t_2 \end{cases}$$

$$P(t_1, t_2, x) = \begin{cases} \frac{(t_1 - x)^2}{2} & \text{if } t_1 > x \\ \frac{(t_2 - x)^2}{2} & \text{if } t_2 > x > t_1 \\ \frac{(1 - x)^2}{2} & \text{if } 1 > x > t_2 \end{cases}$$

Now, considering that an event has a uniform distribution in [0,1], we can derive the expected value for the penalty:

$$\begin{aligned} E[P(t_1, t_2, x)] &= \int_0^{t_1} \frac{(t_1 - x)^2}{2}\, dx + \int_{t_1}^{t_2} \frac{(t_2 - x)^2}{2}\, dx + \int_{t_2}^{1} \frac{(1 - x)^2}{2}\, dx \\ &= \frac{t_1^3}{6} + \frac{(t_2^3 - 3t_1t_2^2 + 3t_1^2t_2 - t_1^3)}{6} - \frac{(t_2^3 - 3t_2^2 + 3t_2 - 1)}{6} \\ &= \frac{(-3t_1t_2^2 + 3t_1^2t_2 + 3t_2^2 - 3t_2 + 1)}{6} = g(t_1, t_2, x) \end{aligned} \tag{2.3}$$

And our goal is to minimize this value, or, to solve an optimization problem:

$$\arg\min_{t_{1,2}} E[P(t_1, t_2, x)], \text{ s.t. } 0 < t_1 < t_2 < 1 \tag{2.4}$$

Figure 2.2 shows how the function looks like on the 3D plane.

Now, we just need to minimize g in two variables. For this we first need to

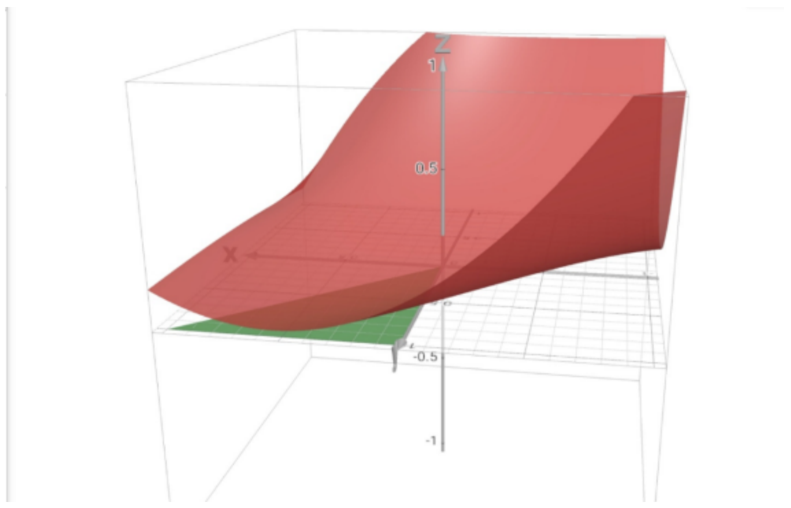

Figure 2.2: Expected value function for two monitoring points

find the gradient of g that is equal to:

$$\nabla g = \begin{bmatrix} \dfrac{-3t_2^2 + 6t_1t_2}{6} \\ \dfrac{-6t_1t_2 + 3t_1^2 + 6t_2 - 3}{6} \end{bmatrix} \tag{2.5}$$

$$\nabla g = 0 \Leftrightarrow \begin{cases} t_2 = 2t_1 \\ t_1 = 2t_2 - 1 \end{cases} \tag{2.6}$$

Where this values were found by solving the quadratic equation:

$$-6t_1t_2 + 3t_1^2 + 6t_2 - 3 = 0 \tag{2.7}$$

$$t_1^2 - 2t_1t_2 + (2t_2 - 1) = 0 \tag{2.8}$$

Solutions for this quadratic equation are:

$$\begin{aligned} t_{1_{1,2}} &= \frac{2t_2 \pm \sqrt{(2t_2-2)^2}}{2} \\ &= \frac{2t_2 \pm (2 - 2t_2)}{2} = \begin{bmatrix} 1, & \text{cannot be since } t_1 < t_2 < 1 \\ 2t_2 - 1 \end{bmatrix} \end{aligned} \tag{2.9}$$

$$t_1 = 4t_1 - 1 => t_1 = \frac{1}{3} \tag{2.10}$$

Finally, we have:

$$\begin{cases} t_1 = \frac{1}{3}, \\ t_2 = \frac{2}{3} \end{cases} \tag{2.11}$$

This is an expected result, as from geometric interpretation of a problem, it corresponds to minimizing the areas of triangles that represent the penalty function. By repeating the calculations with more monitoring points, it can be easily shown that monitoring times should be evenly spaced and they should separate the distribution in equal parts to achieve the minimal total penalty for this distribution.

Next, we will look at a more general statement of this problem.

3 Uniformly Distributed Event With Multiple Monitoring Points

3.1 Problem Statement

Now we consider a more general statement of the problem where we can place an arbitrary number of monitoring points and they are constrained between [0, a], where $a \leq 1$.

- **Objective:** Minimize the expected Age of Information (AoI) for an event that follows uniform distribution over $[0, a]$.
- **Variables:** The locations of k monitoring points, $t_1, t_2, ..., t_k$ where $0 \leq t_1 < t_2 < t_n < t_k \leq 1$.
- **Constraints:** The monitoring points must be placed within the normalized time horizon $[0, 1]$.

This is a more general statement of the problem that allows for more interesting results. In this case, we pay a higher penalty if the event was not detected (monitoring point was placed before an event actually took place) since our linear penalty increases up until the end of the time window, that is until 1. For example, for server monitoring, the interval $[0, a]$ can represent a specific time window during which server activity is of particular interest. Parameter a might correspond to peak hours when server load is highest and the probability of encountering performance issues or failures is greater.

3.2 Derivation

By following similar derivation path as in the chapter 2, we can write the expression for the total penalty as a function of several monitoring points:

$$P(t_1, t_2, ..., t_k, x) = \begin{cases} \frac{(t_1-x)^2}{2} & \text{if } t_1 > x \\ \frac{(t_2-x)^2}{2} & \text{if } t_2 > x > t_1 \\ ... \\ \frac{(t_k-x)^2}{2} & \text{if } t_k > x > t_{k-1} \\ \frac{(1-x)^2}{2} & \text{if } a > x > t_k \end{cases}$$

$$P(t, x) = \begin{cases} \frac{(t-x)^2}{2} & \text{if } t > x \\ \frac{(1-x)^2}{2} & \text{if } a > x > t \end{cases}$$

The PDF for uniform distribution in [a, b] is given by:

$$f(x) = \frac{1}{b-a} \tag{3.1}$$

Now, similarly to the previous case, we want to derive the expected value for the penalty:

$$E[P(t_1, t_2, ..., t_k, x)] = \int_0^{t_1} \frac{(t_1-x)^2}{2a}\,dx + \int_{t_1}^{t_2} \frac{(t_2-x)^2}{2a}\,dx + ... + \int_{t_k}^{a} \frac{(1-x)^2}{2a}\,dx \tag{3.2}$$

And this equality is actually valid only for $t \leq a$ because of the constraints for the penalty function. However, this does not represent a problem since if we place a monitoring point at time t ≥ a, we can always move it to t = a and it will not increase our penalty. This is because the event x is bound to happen in the interval [0, a]. Therefore, we can utilize this formula for constraint optimization provided we only consider its values on the interval [0, a]. Thus, we will also need to check edge value at t = a to find the actual minimum.

For 1 monitoring point, we can easily find its optimal position:

$$\begin{aligned} E[P(t_1,x)] &= \int_0^{t_1} \frac{(t_1-x)^2}{2a}\,dx + \int_{t_1}^{a} \frac{(1-x)^2}{2a}\,dx \\ &= \frac{3t_1^2 - 3t_1 + a^3 - 3a^2 + 3a}{6a} = g(t_1,x) \end{aligned} \tag{3.3}$$

$$\frac{dg}{dt_1} = \frac{6t_1 - 3}{6a} \tag{3.4}$$

$$t_1 = 0.5 \tag{3.5}$$

This seems to be the optimal point. This is the same result from the previous statement. However, for values of a $\leq$ 0.5, this is not admissible. Therefore, noticing that the function is decreasing on the interval [0, 0.5], we conclude that the optimal point is t = a. For larger values of a, we take value t = 0.5 as this is the minimum point of the expectation function. This is an interesting saturation effect that arises due to the minimum of the expectation being outside of the distribution for the event. We can also expect this effect to appear when we consider more than one monitoring point. In this case a saturation value could be shifted due to the change of the minimum point of a new expected value function.

Now, similar approach could be used to understand what happens with two monitoring points:

$$\begin{aligned} E[P(t_1,t_2,x)] &= \int_0^{t_1} \frac{(t_1-x)^2}{2a}\,dx + \int_{t_1}^{t_2} \frac{(t_2-x)^2}{2a} + \int_{t_1}^{a} \frac{(1-x)^2}{2a}\,dx \\ &= \frac{-3t_1t_2^2 + 3t_1^2t_2 - 3t_2^2 + 3t_2 - a^3 + 3a^2 - 3a}{6a} = g(t_1,t_2,x) \end{aligned} \tag{3.6}$$

where expectation is defined for $t_1 \leq t_2 \leq$ a.

This expected total penalty is similar as for the uniform distribution defined on [0,1]. However, this time, t_2 cannot be larger than a (due to previous discussion). Therefore, for values of $a < \frac{2}{3}$, $t_2 = a$. If we substitute value for t_2 inside the equation for the expected total penalty:

$$E[P(t_1,t_2,x)] = \frac{3t_1^2 - 3t_1a - a^2}{6} = h(t_1,x) \tag{3.7}$$

Its derivative is equal to:

$$\frac{dh}{dt_1} = \frac{6t_1 - 3a}{6} \tag{3.8}$$

From here, minimum is achieved at $t_1 = \frac{a}{2}$.

Thus, we have two cases.

1) $a < \frac{2}{3}$: in this case, $t_2 = a$ and $t_1 = a/2$.

2) $a \geq \frac{2}{3}$: in this case, $t_2 = \frac{2}{3}$ and $t_1 = \frac{1}{3}$ (as it was shown for the case of uniform distribution on $[0, 1]$).

This again can be described as a saturation effect. Depending on the value of the penalty, we might want either not to pay 'undetected' penalty at all or it becomes more efficient to pay this 'undetected' penalty sometimes to reduce 'late detection' penalty.

As the number of terms grows, the calculations become more and more tedious. Therefore, numerical simulation could be a good way to address this problem for the number of monitoring points larger than two.

3.3 Numerical Simulation of Monitoring Points Placement

To address the problem of optimal placement of multiple monitoring points for a uniform distribution, a Python-based optimization approach was employed. Here is an explanation of the code and its functionality:

```
num_points = 4
x = Symbol('x')
```

Here we define `num_points`, which specifies the number of monitoring points. A symbolic variable x is created for symbolic integration.

```
def get_total_penalty_func(num_of_points, a):
    t = [Symbol("t" + str(i)) for i in range(1, num_of_points + 1)]
    func = list()
    for i in range(num_of_points):
        func.append((t[i] - x) ** 2 / (2*a))
    func.append((1 - x) ** 2 / (2*a))
    return func
```

The `get_total_penalty_func` function constructs the total penalty function based on the number of monitoring points and parameter `a`. It creates a list of penalty terms for each monitoring point.

```
def get_expected_penalty_func(func, a):
    t = ([Symbol("t" + str(i)) for i in range(1, len(func))])
    t.insert(0, 0)
    t.append(a)
    integrals = list()
    for idx, term in enumerate(func):
        integrals.append(integrate(term, (x, t[idx], t[idx + 1])))
    return integrals
```

The `get_expected_penalty_func` function computes the expected penalty function by integrating the penalty terms over the time intervals between monitoring points.

```
def objective(x, fun):
    keys = ["t" + str(i) for i in range(1, num_points + 1)]
    subs = {key: value for key, value in zip(keys, x)}
    return fun.evalf(subs=subs)
```

The `objective` function evaluates the total expected penalty by substituting the current values of the monitoring points into the symbolic expression.

```
a_pos = np.linspace(0, 1.0, 21)
t_vals = list()
t0 = [0 for i in range(num_points)]

cons = ([{'type': 'ineq', 'fun': lambda x: x[i + 1] - x[0]} for i
            in range(num_points - 1)])
```

We create an array of parameters ′a′ for which we want to find the optimal monitoring points. Constraints ensure that the monitoring points are ordered.

```
for a in a_pos:
    pen_func = get_total_penalty_func(num_points, a)
    e_func = get_expected_penalty_func(pen_func, a)
    res = 0
    for p in e_func:
        res = res + p
    bounds = ((0, a) for i in range(num_points))
    solution = minimize(objective, t0, res, method='SLSQP', bounds=bounds,
    constraints=cons)
    t_vals.append(solution.x)

plt.plot(a_pos, t_vals, 'o')
plt.ylim([0, 1])
plt.title("Optimal value for t with " + str(num_points) + " monitoring points",
fontsize="20")
plt.xlabel("a", fontsize="20")
plt.ylabel("t", fontsize="20")
t_labels = ['t' + str(i) for i in range(1, num_points+1)]
plt.legend(t_labels)
plt.savefig("uniform2_" + str(num_points) + ".png")
plt.show()
```

Finally, the code evaluates the optimal monitoring points across different values of a_param and plots the results, showing how the optimal monitoring points vary with changes in a_param. Notice, that possible monitoring points are also upper-bounded by a. This is because of the discussion at the end of subsection 3.2. Full code can be found in the Appendix.

3.3.1 Numerical Simulation Results

The results for different numbers of monitoring points obtained using this numerical simulation are presented below.

As we can see, monitoring points from Figure 3.1 correspond to the previously derived results. Optimal monitoring time for an event distributed uniformly in [0, a] and with linear penalty applied is chosen as a when $a < 0.5$. And then it is set to 0.5 for $a > 0.5$. (Ignore slight deviations for the points after saturation value has been reached. These are caused by approximations used in the optimizer that allow it to converge faster).

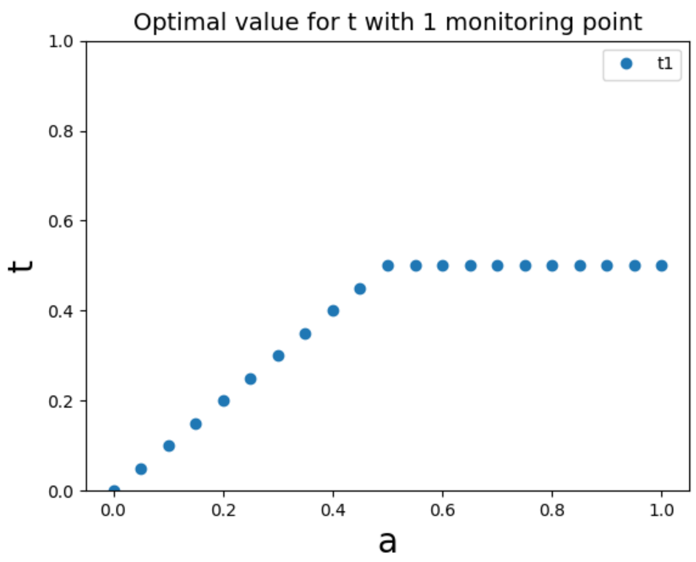

Figure 3.1: Numerical simulation results for 1 monitoring point

For two monitoring points results also correspond. As we can see, the saturation value is larger than before. This was discussed before and it is caused by the fact, that for the uniform distribution [0, 1], optimal monitoring time $t_2 = \frac{2}{3}$. But for values of $a < \frac{2}{3}$, this becomes $t_2 = a$.

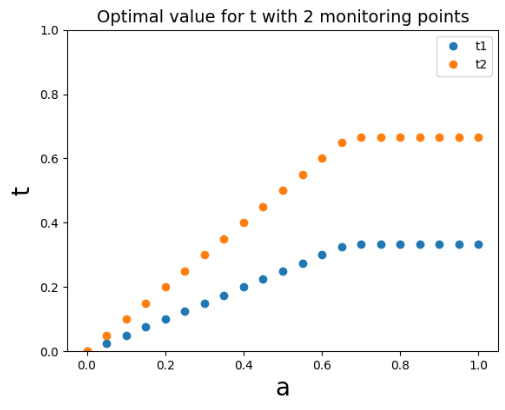

Figure 3.2: Numerical simulation results for 2 monitoring points

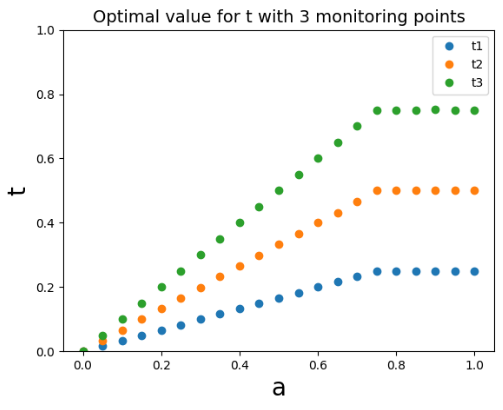

Figure 3.3: Numerical simulation results for 3 monitoring points

Results with a larger number of monitoring points support our conclusions about increasing saturation value for larger number of monitoring points. Also, it shows us that inside the interval [0, a], monitoring points are distributed uniformly. This is evident from the figures for 3, 4, and 10 monitoring points.

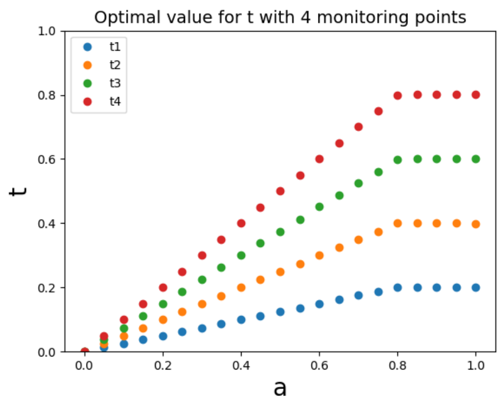

Figure 3.4: Numerical simulation results for 4 monitoring points

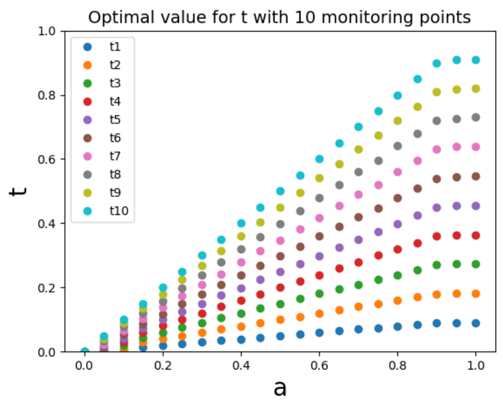

Figure 3.5: Numerical simulation results for 10 monitoring points

4 Gaussian Distribution

We can extend these results to a more sophisticated and realistic case, where an event follows a Gaussian distribution.

- **Objective:** Minimize the expected Age of Information (AoI) for an event that follows Gaussian distribution truncated between [0, 1].
- **Variables:** The locations of k monitoring points, $t_1, t_2, ..., t_k$ where $0 \leq t_1 < t_2 < t_n < t_k \leq 1$.
- **Constraints:** The monitoring points must be placed within the normalized time horizon $[0, 1]$.

We can consider two different distributions. First, we can look at the truncated Gaussian on the range [0, 1] centered at 0.5 with variance equal to 1. Then, we can look at normal distribution also truncated on the range [0, 1]. Expression and derivation for the PDF of a truncated Gaussian distribution can be found in [8].

$$f(x;\mu,\sigma,a,b) = \begin{cases} \frac{\frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right)-\Phi\left(\frac{a-\mu}{\sigma}\right)} & \text{for } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

where:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

is the PDF of the standard normal distribution, and

$$\Phi(z) = \int_{-\infty}^{z} \phi(t)\, dt$$

is the CDF of the standard normal distribution.

Given a Gaussian distribution with mean $\mu = 0.5$ and variance $\sigma^2 = 1$, truncated to the interval $[0, 1]$, the probability density function (PDF) is given by:

$$f(x) = \begin{cases} \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-0.5)^2}{2}}}{\Phi(0.5)-\Phi(-0.5)} & \text{for } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

For now, assume we have only one monitoring point. Thus, total penalty depending on the position of an event and a monitoring point is:

$$P(t, x) = \begin{cases} \frac{(t-x)^2}{2} & \text{if } t > x \\ \frac{(1-x)^2}{2} & \text{if } a > x > t \end{cases}$$

We aim to compute the integral:

$$I = \int_0^{t_1} f(x) \cdot \frac{(t_1 - x)^2}{2}\, dx$$

Substituting $f(x)$ into the integral, we get:

$$I = \int_0^{t_1} \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-0.5)^2}{2}}}{\Phi(0.5) - \Phi(-0.5)} \cdot \frac{(t_1 - x)^2}{2}\, dx$$

This simplifies to:

$$I = \frac{1}{\sqrt{2\pi}(\Phi(0.5) - \Phi(-0.5))} \int_0^{t_1} e^{-\frac{(x-0.5)^2}{2}} \cdot \frac{(t_1 - x)^2}{2}\, dx$$

This is the integral that we get:

$$\int_0^{t_1} e^{-\frac{(x-0.5)^2}{2}} \cdot \frac{(t_1 - x)^2}{2}\, dx$$

Solving this integral by hand is unfeasible, so the optimization was done with the help of numerical simulation.

4.1 Numerical Simulation For Gaussian Distribution

We use similar approach as before, this time we define truncated Gaussian PDF:

```
x = Symbol('x')
u = Symbol('u')

normalization_factor = norm.cdf((b - mu) / sigma) - norm.cdf((a - mu) / sigma)

def gaussian_pdf(x):
    return (1 / (sigma * sqrt(2 * np.pi))) * exp(-0.5 *
    ((x - mu) / sigma) ** 2) / normalization_factor
```

And we also generate an expression for detection function and derive total penalty function:

```
def detection_function(u, x):
    return Abs(u - x)

def get_total_penalty_func(num_of_points):
    t = [Symbol("t" + str(i)) for i in range(1, num_of_points + 1)]
    t.append(b)
    penalty_func = []
    print(t)
    for idx, term in enumerate(t):
        penalty_func.append(integrate(detection_function(term, x), (u, x, t[idx])))
    return penalty_func
```

Finally, we generate expected value function that will later be used for optimization:

```
def get_expected_penalty_func(func):
    integrals = []
    t = ([Symbol("t" + str(i)) for i in range(1, len(func))])
    t.insert(0, a)
    t.append(b)
    for idx, term in enumerate(func):
        integrals.append(integrate(term * gaussian_pdf(x), (x, t[idx], t[idx + 
    return integrals
```

The rest of the script is similar to the one used for uniform distribution. Full code can be found in Appendix 6.2. Mu, sigma, a, b parameters can be adjusted to investigate different Gaussian distributions.

4.1.1 Results from numerical simulation for Gaussian distribution

Results obtained using the script in appendix are presented below. First set of results are for Gaussian distribution with mean 0.5 and variance 1 truncated on the range [0, 1].

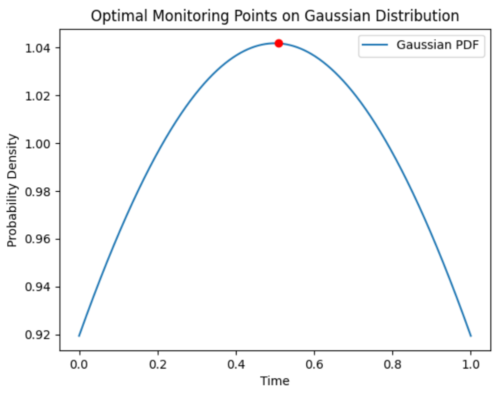

Figure 4.1: Numerical simulation results for Gaussian distribution (0.5, 1) with 1 monitoring point

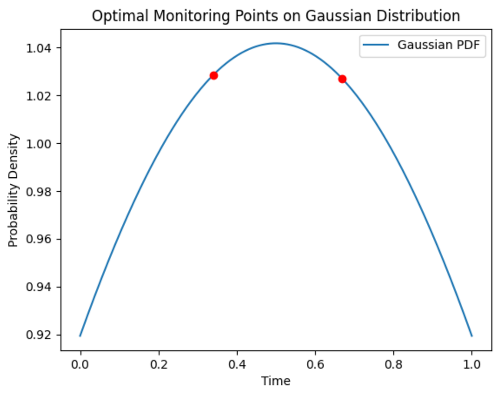

Figure 4.2: Numerical simulation results for Gaussian distribution (0.5, 1) with 2 monitoring points

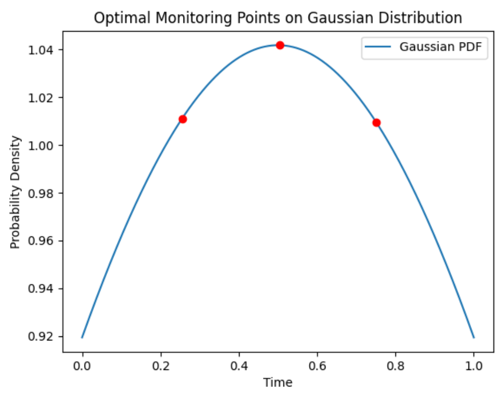

Figure 4.3: Numerical simulation results for Gaussian distribution (0.5, 1) with 3 monitoring points

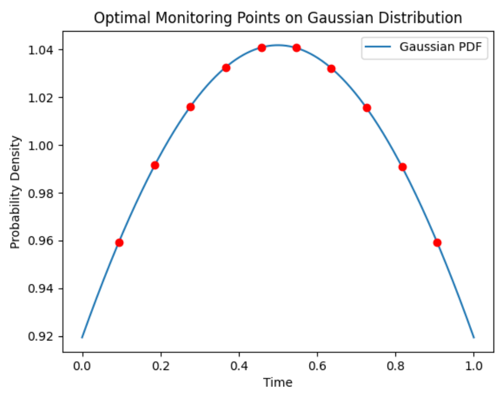

Figure 4.4: Numerical simulation results for Gaussian distribution (0.5, 1) with 10 monitoring points

These results suggest that we aim to place monitoring points symmetrically around the mean. Moreover, these points divide the CDF into equal parts. For example, consider two monitoring points. The optimal placement, according to the simulation, is at t=0.5. This point is where the CDF of the Gaussian distribution is equal to 0.5. For two monitoring points, the positions are approximately $t_1 = 0.33$ and $t_2 = 0.66$. CDF(0.33) ≈ 0.33 and CDF(0.66) ≈ 0.66. This is reminiscent of the uniform distribution case, where points were distributed to divide the uniform distribution into equal parts.

Below, the results for Gaussian distribution with mean 0 and variance 1 truncated on the range [0, 1] are presented:

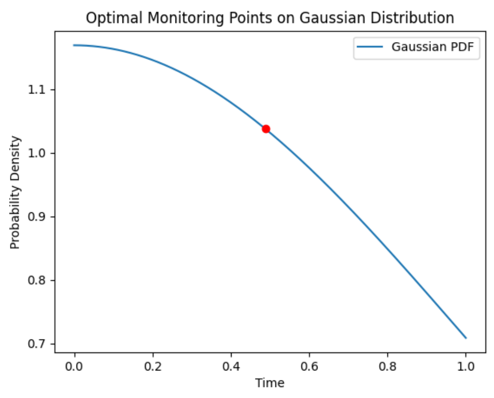

Figure 4.5: Numerical simulation results for Gaussian distribution (0, 1) with 1 monitoring point

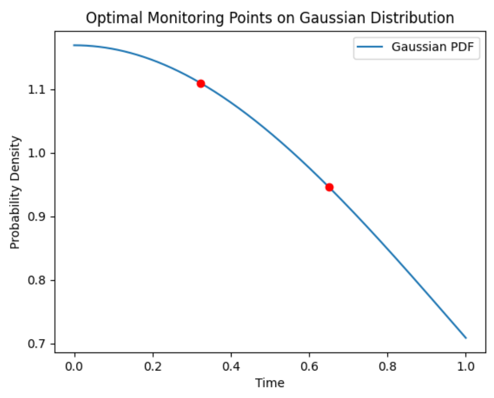

Figure 4.6: Numerical simulation results for Gaussian distribution (0, 1) with 2 monitoring points

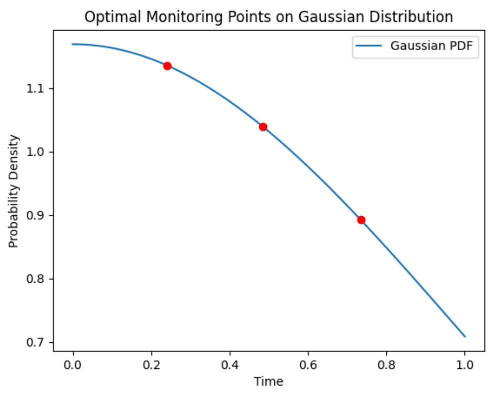

Figure 4.7: Numerical simulation results for Gaussian distribution (0, 1) with 3 monitoring points

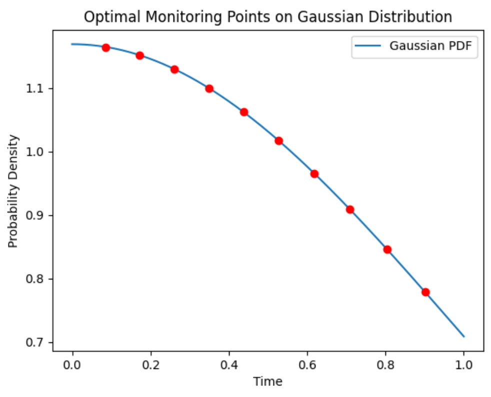

Figure 4.8: Numerical simulation results for Gaussian distribution (0, 1) with 10 monitoring points

For three monitoring points, the script provides the following results:

```
Optimal monitoring points:
t1 = 0.24088284534397714
t2 = 0.4839413806269648
t3 = 0.7348199976076049
```

These are not precise locations as numerical analysis uses a finite number of steps to converge. Thus, this is a close approximation. If we check the CDF at those points, we get approximately 27%, 54%, 78%. If we take into account approximations, this suggests that CDF indeed should be divided in equal parts to minimize the penalty.

5 Conclusions and Future Works

This thesis explored the optimal monitoring point placement strategy to minimize the age of information. This strategy can be used to create and refine update policies for various applications. For example, the time required to complete a computational task on a server or edge device can be modeled using a Gaussian distribution. By estimating this distribution, one can determine optimal instants for checking whether the task has been completed.

In this work, we first explored events that follow a normalized uniform distribution, establishing a theoretical foundation for addressing more complex cases.

Next, a more complex scenario was investigated, where the event follows a uniform distribution over the interval [0,a]. The findings in this case revealed intriguing trends, such as saturation and uniformity in monitoring times, which are evident in the figures from the numerical simulations.

Following this, we examined the Gaussian distribution. This was done with the help of numerical simulations, as the calculations were otherwise unfeasible. In the case of the Gaussian distribution with a mean of 0.5 and variance of 1, the monitoring points were distributed symmetrically around the mean. Moreover, the placement of monitoring points aimed to divide the CDF into equal parts to minimize the total penalty, similar to the uniform distribution case. For the Gaussian distribution with a mean of 0 and variance of 1, the monitoring points were no longer symmetrical. However, they still divided the CDF into equal parts within the truncated range. This result means that further calculations can be simplified by referencing the CDF and placing points so that they divide the CDF into equal parts.

Further works may extend this results by exploring what happens with other distributions and in the case of non-linear penalty function (thus, employing a metric different from AoI). In that case, it is likely that the symmetry will disappear. Moreover, one can explore what happens in the case of discrete distributions.

Another interesting extension would be to create adaptive monitoring system. It would dynamically adjust monitoring points based on incoming data. This would be particularly useful in environments where event distributions can change over time. This can also include cases with unknown distributions that can be learnt over time. Some of the possible approaches to this problem include Approximate Bayesian Computation (ABC) [24] and Kernel Density Estimation (KDE) [25]

One can also aim to minimize two metrics at the same time, i.e. AoI and the cost of monitoring. By considering, for example, energy consumption [21] identified conditions where feedback is advantageous or detrimental depending on feedback costs and channel reliability. Furthermore, one can include impacts of other factors that affect AoI, such as communication, queuing and other delays. This issue is addressed, for instance, in [22].

To conclude, this work provided a brief overview of current studies that involve Age of Information metric and provided a way to optimize update policy based on the event's underlying distribution. This builds an interesting foundation for future studies on optimal timely monitoring of events over a finite time horizon.

6 Appendix

6.1 Code to minimize the penalty function for uniform distribution

```python
import numpy as np
from scipy.optimize import minimize
from sympy import *
import matplotlib.pyplot as plt

num_points = 4
x = Symbol('x')


def get_total_penalty_func(num_of_points, a):
    t = [Symbol("t" + str(i)) for i in range(1, num_of_points + 1)]
    func = list()
    for i in range(num_of_points):
        func.append((t[i] - x) ** 2 / (2 * a))
    func.append((1 - x) ** 2 / (2 * a))
    return func


def get_expected_penalty_func(func, a):
    t = ([Symbol("t" + str(i)) for i in range(1, len(func))])
    t.insert(0, 0)
    t.append(a)
    integrals = list()
    for idx, term in enumerate(func):
        integrals.append(integrate(term, (x, t[idx], t[idx + 1])))
    return integrals


def objective(x, fun):
    keys = ["t" + str(i) for i in range(1, num_points + 1)]
    subs = {key: value for key, value in zip(keys, x)}
    return fun.evalf(subs=subs)


a_pos = np.linspace(0, 1.0, 21)
t_vals = list()
t0 = [0 for i in range(num_points)]
# Constrain each variable, so that t[i+1] > t[i]
cons = ([{'type': 'ineq', 'fun': lambda x: x[i + 1] - x[0]} for i in
    range(num_points - 1)])

for a in a_pos:
    pen_func = get_total_penalty_func(num_points, a)
    e_func = get_expected_penalty_func(pen_func, a)
    res = 0
    for p in e_func:
        res = res + p
    bounds = ((0, a) for i in range(num_points))
    solution = minimize(objective, t0, res, method='SLSQP', bounds=
    bounds, constraints=cons)
    t_vals.append(solution.x)

plt.plot(a_pos, t_vals, 'o')
plt.ylim([0, 1])
plt.title("Optimal value for t with " + str(num_points) + "
    monitoring points", fontsize="20")
plt.xlabel("a", fontsize="20")
plt.ylabel("t", fontsize="20")
t_labels = ['t' + str(i) for i in range(1, num_points + 1)]
plt.legend(t_labels)
plt.savefig("uniform2_" + str(num_points) + ".png")
plt.show()
```

Code 6.1: Python code that produces optimal monitoring points depending on the 'a' parameter and the number of monitoring points

6.2 Code to minimize the penalty function for Gaussian distribution

```python
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from scipy.stats import norm
from sympy import Symbol, integrate, exp, sqrt, pi, Abs

# Define parameters for the Gaussian distribution
mu = 0
sigma = 1
a = 0
b = 1
num_points = 10
x = Symbol('x')
u = Symbol('u')

normalization_factor = norm.cdf((b - mu) / sigma) - norm.cdf((a - mu)
     / sigma)


def gaussian_pdf(x):
    return (1 / (sigma * sqrt(2 * np.pi))) * exp(-0.5 * ((x - mu) /
    sigma) ** 2) / normalization_factor


def gaussian_standard_pdf(x):
    return (1 / (sigma * sqrt(2 * pi))) * exp(-0.5 * ((x - mu) /
    sigma) ** 2)


def detection_function(u, x):
    return Abs(u - x)


def get_total_penalty_func(num_of_points):
    t = [Symbol("t" + str(i)) for i in range(1, num_of_points + 1)]
    t.append(b)
    penalty_func = []
    print(t)
    for idx, term in enumerate(t):
        penalty_func.append(integrate(detection_function(term, x), (u, x, t[idx])))
    return penalty_func


def get_expected_penalty_func(func):
    integrals = []
    t = ([Symbol("t" + str(i)) for i in range(1, len(func))])
    t.insert(0, a)
    t.append(b)
    for idx, term in enumerate(func):
        integrals.append(integrate(term * gaussian_pdf(x), (x, t[idx], t[idx + 1])))
    return integrals


def objective(t_values):
    keys = ["t" + str(i) for i in range(1, num_points + 1)]
    subs = {key: value for key, value in zip(keys, t_values)}
    return sum([penalty.evalf(subs=subs) for penalty in total_expected_penalty])


penalty_func = get_total_penalty_func(num_points)
total_expected_penalty = get_expected_penalty_func(penalty_func)

t0 = [0 for _ in range(num_points)]

# Bounds for t: each t should be within [t_min, t_max]
bounds = [(a, b) for _ in range(num_points)]

# Constraints to ensure t[i] < t[i+1]
constraints = [{'type': 'ineq', 'fun': lambda t: t[i + 1] - t[i]} for i in range(num_points - 1)]

solution = minimize(objective, t0, method='SLSQP', bounds=bounds, constraints=constraints)
optimal_points = solution.x
print("Final value " + str(objective(optimal_points)))

print('Optimal monitoring points:')
for i, t in enumerate(optimal_points, 1):
    print(f't{i} = {t}')

# Plotting the Gaussian PDF and optimal monitoring points
x_vals = np.linspace(a, b, 1000)
y_vals = [float(gaussian_pdf(val)) for val in x_vals]

plt.plot(x_vals, y_vals, label='Gaussian PDF')
plt.scatter(optimal_points, [float(gaussian_pdf(val)) for val in
    optimal_points], color='red', zorder=5)
plt.title('Optimal Monitoring Points on Gaussian Distribution')
plt.xlabel('Time')
plt.ylabel('Probability Density')
plt.legend()
plt.savefig("gaussian1_" + str(num_points) + ".png")
plt.show()
```

Code 6.2: Python code that produces optimal monitoring points for Gaussian distribution depending on it's parameters and number of monitoring points

Bibliography

[1] Mohammed Albekairi. “A Comprehensive Mutable Analytics Approach to Distinguish Sensor Data on the Internet of Underwater Things”. In: *IEEE Access* (2024).

[2] Leonardo Badia and Alessandro Gandelli. “A Comparison of Status Update Optimization and Microgrid Management”. In: *Proc. CGEE*. 2023, pp. 6–10.

[3] Murat Bastopcu and Sennur Ulukus. “Age of information with soft updates”. In: *IEEE Transactions on Information Theory* 66.6 (2020), pp. 3864–3876.

[4] Ahmed M Bedewy, Yifan Sun, and Ness B Shroff. “Optimizing data freshness, throughput, and delay in multi-server information-update systems”. In: *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2016, pp. 2569–2573.

[5] Alessandro Buratto, Hogler Tuwei, and Leonardo Badia. “Optimizing sensor data transmission in collaborative multi-sensor environments”. In: *Proc. IEEE COMNETSAT*. 2023, pp. 635–639.

[6] Alessandro Buratto, Begüm Yivli, and Leonardo Badia. “Machine learning misclassification within status update optimization”. In: *Proc. IEEE COMNETSAT*. 2023, pp. 640–645.

[7] Alessandro Buratto et al. “Game theoretic analysis of AoI efficiency for participatory and federated data ecosystems”. In: *Proc. IEEE ICC Workshops*. 2023, pp. 1301–1306.

[8] John Burkardt. *The Truncated Normal Distribution*. `https://people.sc.fsu.edu/~jburkardt/presentations/truncated_normal.pdf`. Department of Scientific Computing Website, Florida State University, Tallahassee. 2018.

[9] Kuan Chen and Longbo Huang. “Age-of-information in the presence of error”. In: *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2016, pp. 2571–2575.

[10] Michele Costa, Mihai Codreanu, and Anthony Ephremides. “Age of information with packet management”. In: *2014 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2014, pp. 1583–1587.

[11] Josefine Holm et al. “Goal-oriented scheduling in sensor networks with application timing awareness”. In: *IEEE Trans. Commun.* 71.8 (2023), pp. 4513–4527.

[12] Yi-Pin Hsu, Eytan Modiano, and Lingjie Duan. “Scheduling algorithms for minimizing age of information in wireless networks with stochastic arrivals”. In: *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2017, pp. 2563–2567.

[13] Yung-Lin Hsu et al. “Age-optimal power allocation in industrial IoT: A risk-sensitive federated learning approach”. In: *Proc. IEEE PIMRC*. 2021, pp. 1323–1328.

[14] Alireza Javani, Marwen Zorgui, and Zhiying Wang. “On the age of information in erasure channels with feedback”. In: *Proc. IEEE ICC*. 2020.

[15] Zhuoxuan Ju, Parisa Rafiee, and Omur Ozel. “Optimizing urgency of information through resource constrained joint sensing and transmission”. In: *Entropy* 24.11 (2022), p. 1624.

[16] Chee Yeun Kam et al. “Age of information with a packet deadline”. In: *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2016, pp. 2569–2573.

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

[18] Sanjit Kaul, Roy D Yates, and Marco Gruteser. “Status updates through queues”. In: *2012 46th Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2012, pp. 1–6.

[19] Ali Maatouk et al. “The age of information in a discrete time queue: Stationary distribution and non-linear age functions”. In: *IEEE Transactions on Information Theory* 66.3 (2020), pp. 1784–1801.

[20] Eytan Modiano. “Minimizing age of information in wireless networks with stochastic arrivals”. In: *Proceedings IEEE INFOCOM*. IEEE. 2019, pp. 173–181.

[21] Andrea Munari and Leonardo Badia. “The role of feedback in AoI optimization under limited transmission opportunities”. In: *Proc. IEEE Globecom*. 2022, pp. 1972–1977.

[22] Devarpita Sinha and Rajarshi Roy. “Scheduling Status Update for Optimizing Age of Information in the Context of Industrial Cyber-Physical System”. In: *IEEE Access* 7 (2019), pp. 95677–95695. DOI: 10.1109/ACCESS.2019.2919320.

[23] Yifan Sun et al. “Update or wait: How to keep your data fresh”. In: *IEEE Transactions on Information Theory* 63.11 (2017), pp. 7492–7508.

[24] Mikael Sunnåker et al. “Approximate Bayesian computation”. en. In: *PLoS Comput. Biol.* 9.1 (Jan. 2013), e1002803.

[25] Stanislaw Weglarczyk. “Kernel density estimation and its application”. In: *ITM Web of Conferences* 23 (Nov. 2018), p. 00037. DOI: 10.1051/itmconf/20182300037.

[26] Roy D Yates and Sanjit Kaul. “Real-time status updating: Multiple sources”. In: *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE. 2012, pp. 2666–2670.

[27] Roy D. Yates et al. “Age of Information: An Introduction and Survey”. In: *IEEE Journal on Selected Areas in Communications* 39.5 (2021), pp. 1183–1210. DOI: 10.1109/JSAC.2021.3065072.

[28] Yeşim Yiğitbaşı, Fabio Stroppa, and Leonardo Badia. “Optimizing Real-Time Decision-Making in Sensor Networks”. In: *2023 16th International Conference on Developments in eSystems Engineering (DeSE)*. 2023, pp. 206–211.

[29] Chen Ying et al. “An AoTI-driven joint sampling frequency and access selection optimization for industrial wireless sensor networks”. In: *IEEE Trans. Veh.Techn.* 72.9 (2023), pp. 12311–12325.

[30] Alberto Zancanaro, Giulia Cisotto, and Leonardo Badia. “Tackling age of information in access policies for sensing ecosystems”. In: *Sensors* 23.7 (2023), p. 3456.

[31] Alberto Zancanaro et al. "Impact of transmission delays over age of information under finite horizon scheduling". In: *Proc. IEEE CAMAD*. 2023, pp. 99–104.