AoI Minimization in Single Source Node System

411086010 通訊三 林紘毅

https://github.com/linnnz19/Queueing-Theory-Project-Program

1. Introduction

In recent years, researchers in the area of time-sensitive applications are interested in maintaining information freshness. Information freshness can be captured by a new metric called Age of Information (AoI), which is first introduced in [1]. AoI is defined as the time elapsed since the generation of the status update that was most recently received by a destination.

This report simulates an AoI minimization problem using a backward dynamic programming algorithm. We reference the concepts and system outlined in [2] and design our own version of a time-correlated model. This report aims to solve this simplified version of a Markov Decision Problem using the high-level concepts of a dynamic programming algorithm, which can provide us with the optimal policies.

2. System Model

We considered a system including single source node that generates status update packets, which need to be transmitted to a destination node (monitor) over a wireless channel.

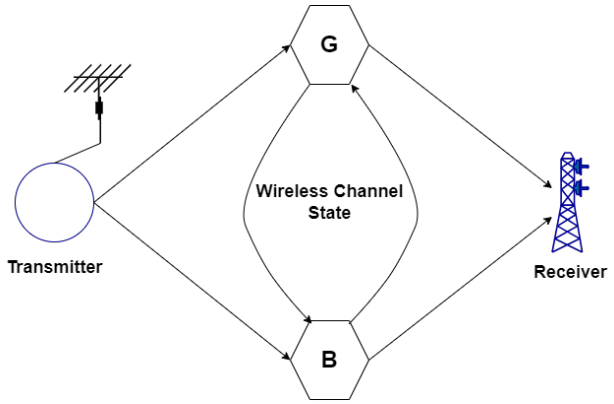

Figure 2.1: System with single source node and single destination node

The goal is to minimize the time-average AoI at the monitor by optimally scheduling the transmissions from the source. We directly formulated this problem into a Markov Decision Problem, and the system model can be defined as follows:

1. Time is divided into discrete time slots, indexed by t = 0, 1, 2, ..., T, where T is the time horizon over which we want to minimize the AoI.

2. At each time slot, the source node can either transmit a single status update packet or remain idle.

3. The wireless channel is assumed to be a time-correlated fading channel and evolve as a two-state Gilbert-Elliot channel, which the two states are: Good (G) and Bad (B). The channel state transitions between G and B according to a Markov chain with transition probabilities:

   - $P(G \longrightarrow G) = p$
   - $P(G \longrightarrow B) = 1 - p$
   - $P(B \longrightarrow G) = q$
   - $P(B \longrightarrow B) = 1 - q$

   This means that the channel state at the current time slot depends on the channel state in the previous time slot, introducing temporal correlation and memory into the channel model.

4. If the source transmits a packet in state G, the packet is successfully received at the monitor with probability 1. If the source transmits a packet in state B, the packet is successfully received with probability ε, where $0 < \varepsilon < 1$.

5. The AoI is reset to 1 upon receiving a new update and incremented by 1 for each time slot that passes without a successful reception.

6. The cost function is the time-average AoI over the time horizon T, which needs to be minimized.

3. Problem Formulation

In this section, we provide the formulation of the problem. We aim to minimize the time-average AoI at the monitor by optimally scheduling transmissions from the source node. We slackly assume that our problem is a CMDP with state s(t) = (t, a, c), where:

- t is the current time slot
- a is the current AoI at the monitor
- c is the current channel state (G or B)

Let $V_t(s_t)$ denote the minimum expected cost (AoI) starting from state s_t at time t to the end of the horizon T. The goal is to determine the optimal policy that minimizes this expected cost. In this way, the Bellman's equation can be described below:

$$V_t(s_t) = \min_{u_t \in \{0,1\}} \left\{ C(s_t, u_t) + \mathbb{E}\left[V_{t+1}(s_{t+1}) \mid s_t, u_t \right] \right\} \tag{3.1}$$

- u_t is the action taken at time t (0 for idle, 1 for transmit)
- $C(s_t, u_t)$ is the immediate cost incurred in state s_t when action u_t is taken.
- $E\left[V_{t+1}(s_{t+1}) | s_t, u_t \right]$ is the expected cost-to-go from state s_{t+1} at time $t+1$, given that u_t was taken in state s_t.

The immediate cost $C(s_t, u_t)$ is simply the AoI at the monitor, which is a.

The next state $s_{t+1} = (t+1, a_{t+1}, c_{t+1})$ depends on the current state s_t and the action u_t as follows:

1. If $u_t = 0$ (remain idle):
   - The AoI increases by 1: $a_{t+1} = a + 1$
   - The channel state transitions according to the Markov chain probabilities, which is already provided in system model.
2. If $u_t = 1$ (transmit a status update packet):
   - If the channel state is Good (G), the transmission successful with probability 1:
     - The AoI resets to 1: $a_{t+1} = 1$
   - If the channel state is Bad (B), the transmission successful with probability ε and fails with probability $1 - \varepsilon$
     - Successful transmission: $a_{t+1} = 1$
     - Failed transmission: $a_{t+1} = a + 1$

Thus, the Bellman's equation can be explicitly written as:

$$\begin{aligned} V_t(s_t) = \min \Bigg\{ & a + \mathbb{E}\left[V_{t+1}(t+1, a+1, c_{t+1}) \mid s_t, u_t = 0 \right], \\ & a + \epsilon \mathbb{E}\left[V_{t+1}(t+1, 1, c_{t+1}) \mid s_t, u_t = 1, \text{success} \right] \\ & + (1-\epsilon) \mathbb{E}\left[V_{t+1}(t+1, a+1, c_{t+1}) \mid s_t, u_t = 1, \text{failure} \right] \Bigg\} \end{aligned} \tag{3.2}$$

4. Solution of the MDP

For this finite-horizon MDP problem, we can use the similar mathematical technique described in [2] to solve the recursion in (3.2) by using backward dynamic programming. The high-level pseudo code of this algorithm is described as below:

Algorithm 1 Backward Dynamic Programming for AoI Minimization

For $t = T$ **to** 0:
  For each state $s = (t, a, c)$ **in** S:
    If $t == T$:
      $V(s) = a$
    Else:
      $min\_cost = \infty$
      $best\_action =$ None
      For each action a:
        $cost = a$ ▷ Immediate cost
        For each possible next state $s' = (t+1, a', c')$:
          $p =$ TransitionProbability(s, a, s')
          $cost+ = p * V(s')$
      If $cost < min\_cost$:
        $min\_cost = cost$
        $best\_action = a$
    $V(s) = min\_cost$
    $\pi(s) = best\_action$
Return V, π

In this algorithm, it estimates the value function for each state at every slot by proceeding backward in time. Thus, for each scheduling decision, the algorithm calculates the value function for every possible s' state. Ultimately, we can get the best action at each state. With the computed optimal policy through every frame, we can calculate the average AoI in whole system easily. Pseudo code of calculating average AoI is provided as below:

```
Initialize state s = (0, 1, 'G')
total_aoi = 0
For t = 0 to T:
    total_aoi += a // Current AoI
    action = π(s)
    s' = GetNextState(s, action) // Based on transition probabilities
    s = s'
average_aoi = total_aoi / (T+1)
Return average_aoi
```

5. Analysis of Simulation Results

In this section, we provide several simulation results and observations of the time average AoI. For the following results, we consider the Gilbert-Eilliot channel, with $p = 0.8$, $q = 0.6$, and $\varepsilon = 0.2$. The length of the time horizon, T, is 30 time slots. We run 10000 times of simulations and observe the distribution of the average AoI:

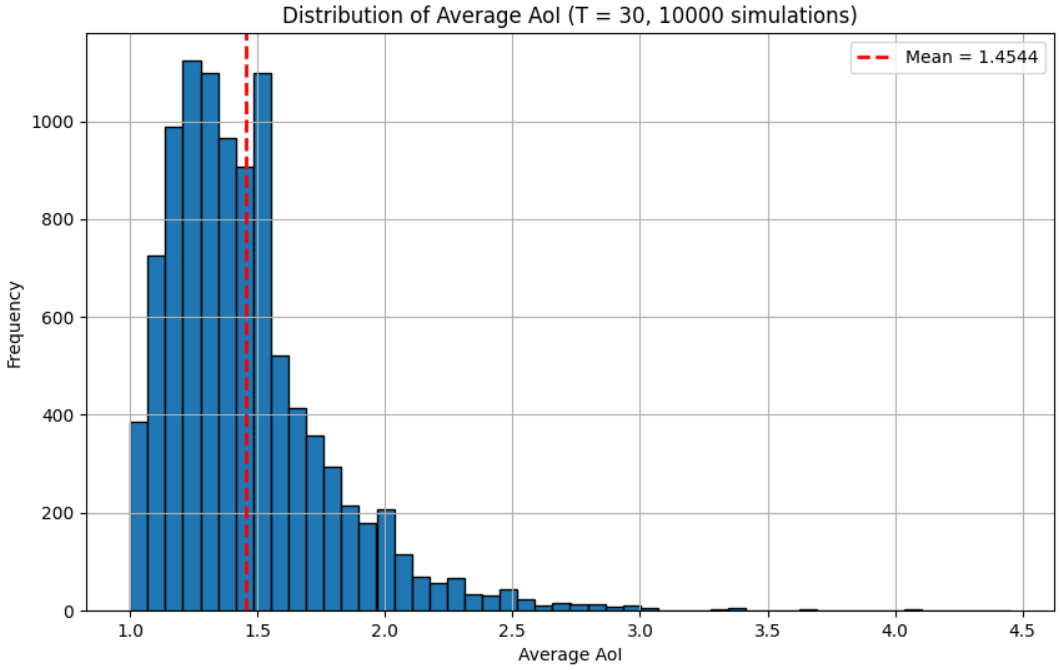

Figure 5.1: Distribution of Average AoI

In Figure (5.1), we provide the distribution of average AoI and also calculate the mean value. Here are several observations of this result:

- Distribution Shape: The distribution exhibits a right-skewed (positive-skewed) pattern. Most values are concentrated to the left of the mean, with a longer tail extending towards the right.
- Peak: The mode of the distribution is slightly to the left of the mean, approximately between 1.3 and 1.4.
- Continuity: The distribution appears continuous, with no obvious gaps or anomalies.
- Unimodal distribution: There is only one clear peak, indicating a unimodal distribution.

With this observation, we slackly assumed that this distribution pattern is likely a Gamma distribution or a Log-normal distribution. This kind of distribution pattern is also reasonable in AoI systems because:

1. Most of the time, the system can maintain relatively low AoI (values close to 1).

2. Occasionally, longer update intervals occur, leading to higher AoI values, forming the long tail on the right.

Overall, this distribution tells us that while the system performs well most of the time (maintaining low AoI), there's still a probability of higher AoI values occurring, which may be due to the randomness of the communication channel or other system factors. In my opinion, this characteristic make sense to a nearly stable system.

On the other hand, I also wrote a program that can determine the optimal policy for each corresponding state. With this program, we can calculate optimal policies under any set of environmental parameters. Although this is a detailed aspect and not the main result of this report, I still believe it is vital for future research. Therefore, I have decided to share the optimization policies' detail below, you are also more than welcome to check the program *optimal_policy.py* in attachment. (Noticed that the AoI value has been restrict not to be greater than 10 referencing [2]):

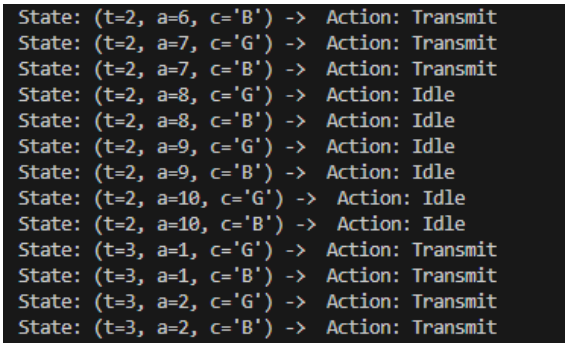

Figure 5.2: Optimal Policy under each state

Lastly, we want to take a look into convergence of this algorithm. In Figure (5.3), (5.4) we provide the different values of T to observe whether the algorithm converges to a stable average AoI as the time horizon increases.

Observing these patterns, we can conclude that:

1. The algorithm converges to a stable average AoI as the time horizon increases.

2. The rate of change in average AoI decreases for larger T values, further supporting convergence.

This may not be a formal mathematical proof, but it provides strong empirical evidence of the algorithm's convergence. The convergence indicates that the optimal policy stabilizes for sufficiently large T, leading the system to reach a steady-state behavior.

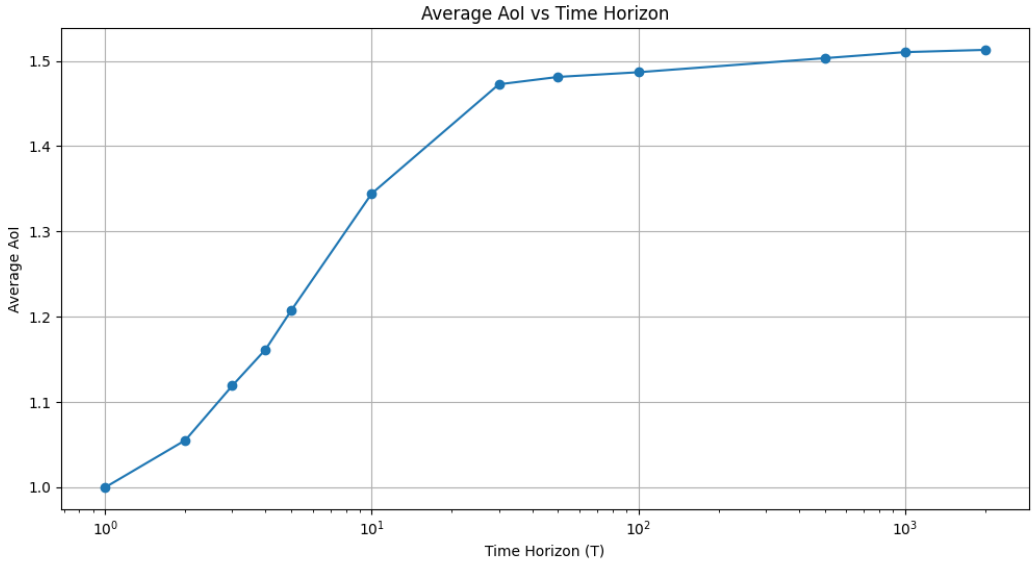

Figure 5.3: Convergence test with T from 1 to 2000

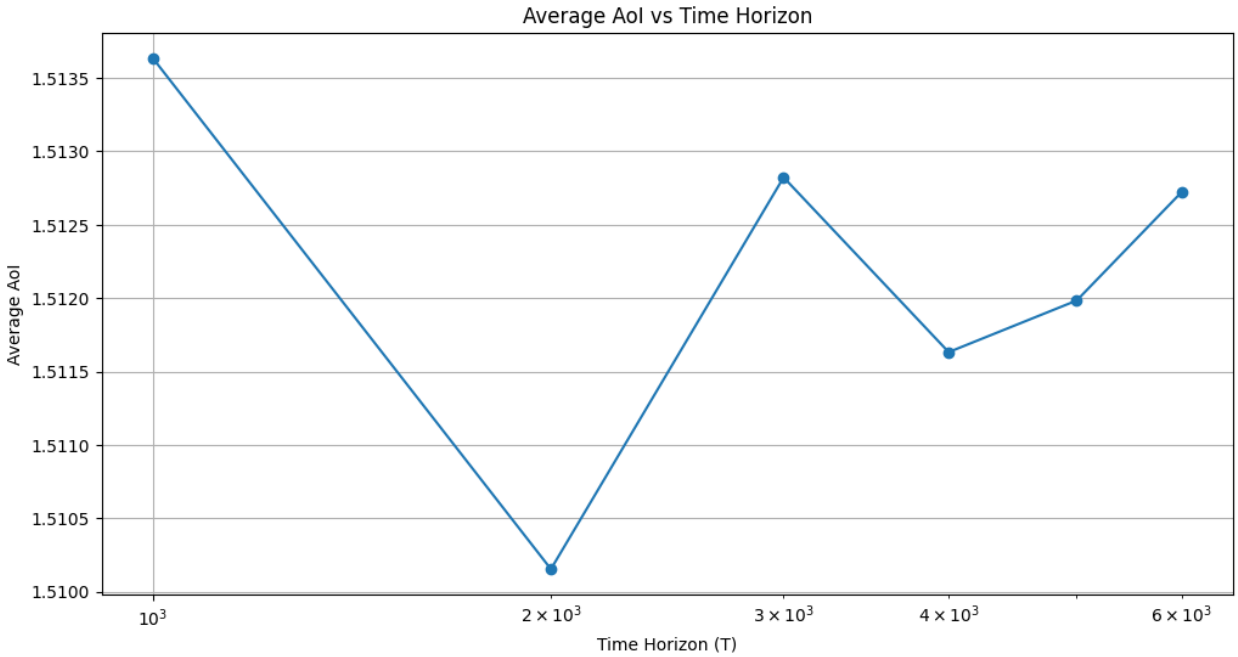

Figure 5.4: Convergence test with T from 1000 to 6000

6. Conclusions

Based on our analysis and the results, here's an overall conclusion of this work:

1. **Distribution histogram:** The distribution provides valuable insights into the system's behavior in maintaining information freshness, showing both its typical performance and the range of possible outcomes across 10000 simulations.

2. **Impact of time horizon:** The choice of time horizon T significantly affects the computed optimal policy and resulting average AoI for smaller T values. However, as T increases, the impact diminishes, and the system's performance stabilizes.

3. **Convergence of the algorithm:** We've demonstrated empirically that the backward dynamic programming algorithm converges as the time horizon T increases. The average AoI stabilizes for larger T values, indicating that the system reaches a steady-state behavior.

4. **Insights for system design:** The results provide valuable insights for designing communication systems with AoI constraints. The optimal policies derived from this method can guide the development of practical transmission strategies that maintain information freshness in the presence of unreliable channels.

7. Extensions and future work

This study focused on a specific channel model and system setup, which is relatively simple one contrast to [2]. In future work, we could explore more complex channel models, multi-source scenarios, or even the incorporation of energy constraints. Additionally, extending the analysis to infinite horizon problems or investigating the theoretical bounds on AoI performance may provide further insights.

References

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

[2] E. Fountoulakis, T. Charalambous, A. Ephremides and N. Pappas, "Scheduling policies for AoI minimization with timely throughput constraints", *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 3905-3917, Jul. 2023.