Time-, Graph- and Value-based Sampling of Internet of Things Sensor Networks

Holm, Josefine

DOI (link to publication from Publisher):
10.54337/aau532689661

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

TIME-, GRAPH- AND VALUE-BASED SAMPLING OF INTERNET OF THINGS SENSOR NETWORKS

BY
JOSEFINE HOLM

DISSERTATION SUBMITTED 2023

AALBORG UNIVERSITY
DENMARK

Time-, Graph- and Value-based Sampling of Internet of Things Sensor Networks

Ph.D. Dissertation

Josefine Holm

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

Dissertation submitted: February 2023

PhD supervisor: Professor Petar Popovski
Aaqlborg University

PhD committee: Associate Professor Tatiana Kozlova Madsen (chair)
Department of Electronic Systems
Aalborg University

Associate Professor Jemin Lee
Department of Electronic and Electrical Engineering
Sungkyunkwan University (SKKU)
Suwon, Korea

Professor Sofie Pollin
TELEMIC
KU Leuven, Belgium

PhD Series: Technical Faculty of IT and Design, Aalborg University

Department: Department of Electronic Systems

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-735-2

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Stibo Complete, 2023

Abstract

The rise of 5G and Internet of Things (IoT) technology has brought the option to collect massive amounts of data. However, there are limitations imposed by bandwidth, storage space, and energy consumption and much work is being done to mitigate and loosen these limitations. However, once a hardware solution has been installed, the limit is set and it can be costly to replace it. When the limit is reached, the way forward is to decrease the volume of data by increasing its value or omitting the data with little to no value. The goal is then to devise clever sampling and transmission schemes for collecting only the most valuable data. Value of Information (VoI) is a measure of how useful a data point is, with three important aspects: is it collected at the right time, is it relevant for the problem at hand, and is it providing novelty that could not be predicted?

This thesis explores the use of side information for optimizing Value of Information (VoI) in a wireless IoT network setting. Multiple types of side information are considered in order to optimize the different aspects of VoI. First, graph structures are used to increase the novelty of data transmitted. Second, knowledge of the query arrival process is used to minimize the response time. Third, knowledge of the aggregation functions used to summarize the data is used to maximize the relevance of the data collected. The results show that it is possible to tailor the sampling and transmission scheme to a given application using side information. It is further shown that the more precise, extensive, and diverse the side information available is, the better the results are. Partial side information can also be useful, though with diminishing returns. A secondary focus of the thesis is to make the transmission scheme more efficient and increase the lifetime of battery driven IoT devices. This is done by lowering the total number of transmissions necessary and by forcing the optimizations algorithms to adhere to a strict transmission budget.

Resumé

Fremkomsten af 5G og Internet of Things (IoT) teknologi har medført muligheden for at indsamle enorme mængder data. Det er dog begrænset af båndbredde, lagerplads og energiforbrug og meget arbejde bliver gjort for at afbøde og løsne disse begrænsninger. Der arbejdes meget på at øge disse, men når først en hardwareløsning er installeret, er grænsen fastsat, og det kan være dyrt at udskifte den. Når grænsen er nået, er vejen frem at reducere mængden af data, ved at øge dets værdi eller udelade det med ringe eller ingen værdi. Målet er derefter at udtænke intelligente sampling- og transmissionsplaner til kun at indsamle de mest værdifulde data. Value of Information (VoI) er et mål for, hvor nyttigt et datapunkt er, med tre vigtige aspekter: er det indsamlet på det rigtige tidspunkt, er det relevant for det aktuelle problem og bidranger det med noget nyt some ikke kunne forudsiges?

Denne afhandling udforsker brugen af sideinformation til optimering af VoI i en trådløs IoT-netværksforbindelse. Flere typer sideinformation overvejes for at optimere de forskellige aspekter af VoI. For det første bruges grafstrukturer til at øge nyhedsværdien af de transmitterede data. For det andet bruges kendskab til forespørgselsankomstprocessen til at minimere responstiden. For det tredje bruges viden om de aggregeringsfunktioner, der bruges til at opsummere dataene, for at maksimere relevansen af de indsamlede data. Resultaterne viser, at det er muligt at skræddersy sampling- og transmissionsplaner til en given anvendelse ved hjælp af sideinformation. Det er yderligere vist, at jo mere præcis, omfattende og forskelligartet den tilgængelige sideinformation er, jo bedre er resultaterne. Delvis sideinformation kan også være nyttig, dog med aftagende afkast. Et sekundært fokus i afhandlingen er at gøre transmissionsplanerne mere effektive, og øge levetiden for batteridrevne IoT-enheder. Dette gøres ved at sænke det samlede antal nødvendige transmissioner, og ved at tvinge optimeringsalgoritmerne til at overholde et stramt transmissionsbudget.

Contents

Thesis Details

Thesis Title:	Time-, Graph- and Value-based Sampling of Internet of Things Sensor Networks
Ph.D. Student:	Josefine Holm
Supervisors:	Prof. Petar Popovski, Aalborg University
Co-Supervisors:	Prof. Morten Nielsen, Aalborg University Asst. Prof. Federico Chiariotti, Aalborg University

The main body of this thesis consist of the following papers.

[A] Josefine Holm, Federico Chiariotti, Morten Nielsen, and Petar Popovski, "Lifetime Maximization of an Internet of Things (IoT) Network based on Graph Signal Processing," *IEEE Communications Letters*, Vol. 25, No. 8, pp. 2763–2767, 2021.

[B] Roshni Chakraborty, Josefine Holm, Torben Bach Pedersen, and Petar Popovski, "Finding Representative Sampling Subsets in Sensor Graphs using Time Series Similarities," Submitted to *ACM Transactions on Sensor Networks*, 2022.
Preprint: `https://arxiv.org/pdf/2202.08504.pdf`

[C] Josefine Holm, Anders E. Kalør, Federico Chiariotti, Beatriz Soret, Søren K. Jensen, Torben B. Pedersen, and Petar Popovski, "Freshness on Demand: Optimizing Age of Information for the Query Process," *IEEE International Conference on Communications*, pp. 1–6, 2021.

[D] Federico Chiariotti, Josefine Holm, Anders E. Kalør, Beatriz Soret, Søren K. Jensen, Torben B. Pedersen, and Petar Popovski, "Query Age of Information: Freshness in Pull-Based Communication," *IEEE Transactions on Communications*, vol. 70, no. 3, pp. 1606–1622, 2022.

[E] Søren Kejser Jensen, Josefine Holm, Federico Chiariotti, Christian Thomsen, Anders Ellersgaard Kalør, Petar Popovski, Beatriz Soret, and Torben Bach Pedersen, "SENDAI: A Framework for Joint Reasoning About Sensor Data Acquisition and Sensor Data Analytics," Submitted to *Information and Computation Journal*, 2022.

[F] Federico Chiariotti, Anders E. Kalør, Josefine Holm, Beatriz Soret, and Petar Popovski, "Scheduling of Sensor Transmissions Based on Value of Information for Summary Statistics," *IEEE Networking Letters*, vol. 4, no. 2, pp. 92–96, 2022.

[G] Josefine Holm, Federico Chiariotti, Anders E. Kalør, Beatriz Soret, Torben B. Pedersen, and Petar Popovski, "Goal-Oriented Scheduling in Sensor Networks with Application Timing Awareness," Submitted to *IEEE Transactions on Communications*, 2022.

In addition to the main papers, the following publications have also been made.

[I] Josefine Holm, Thomas Arildsen, Morten Nielsen, and Steffen Lønsmann Nielsen,, "Orthonormal, moment preserving boundary wavelet scaling functions in Python," *SN Applied Sciences*, vol. 2, no. 12, pp. 1–9, 2020.

[II] Ivana Nikoloska, Josefine Holm, Anders E. Kalør, Petar Popovski, Nikola Zlatanov, "Inference over Wireless IoT Links with Importance-Filtered Updates," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1089–1098, 2021.

[III] Josefine Holm, Alberto Natali, Geert Leus, and Petar Popovski, "Reconstruction of Missing Values in Sub-Sampled IoT Sensor Networks through Graph Signal Processing," in preparation.

Acknowledgements

I started my PhD with a great curiosity to explore the dept of this rabbit hole and to push the limits of my own abilities.

I would like like to thank my supervisors - Professor Petar Popovski, for you tremendous help and support and for making time and space for me, not just as your student but as a person; Professor Morten Nielsen who is the reason that I became interested in research in the first place; Assistant professor Federico Chiariotti for always having your door open to my questions and for helping turn around projects that I had given up on.

Secondly I would like to thank my fellow researchers and technical staff in the connectivity section at Aalborg University as well as all of my co-authors. Especially those from the departments of mathematical and computer science, as interdisciplinary work has always been a special interest of mine.

I would like to thank Geert Leus and his fellow researchers at TU Delft for hosting me and including me in the day-to-day work and scocial life of the group. I would also like to thank the assessment committee for evaluating my work as well as the sources of funding that have made this possible.

Lastly I would like to thank my friends and family, specially my partner Peter, who has been my rock and my cheerleader though all the ups and downs of the past few years, I will be forever grateful for you support.

Josefine Holm
Aalborg University, February 28, 2023

Part I

Introduction

1 Introduction

The introduction of 5G brought with it new ways of viewing wireless connectivity. Previous generations aimed at the highest possible capacity as the first and foremost objective, while 5G broadens the scope to new performance metrics for different types of traffic. In addition to Enhanced Mobile Broadband (eMBB), which is an extension of traditional mobile broadband, Ultra Reliable Low Latency Communications (URLLC) and Massive Machine Type Communications (mMTC) were introduced [20]. These are terms for three different categories of use cases; eMBB is considering relatively few users who need a high data rate without particularly strict requirements on latency and reliability [8]. One common use-case in this category is people streaming video on a wireless device. URLLC considers problems where it is vital that a small amount of data from few users is received correctly without delay. One example use-case in this category is self driving cars where missing or delayed information can cause a crash. mMTC considers a high number of users needing to transmit small amounts of data without strict requirements on latency and reliability. One common case is small devices observing the environment, such as the temperature, and reporting this to a central system.

The focus of this thesis is mMTC use cases, i.e., many small devices reporting observations of a process or their environment to a central point. Networks of Internet of Things (IoT) devices are becoming more and more common, making it an important area for research. The characteristics of IoT devices are:

- Very cheap to install in high numbers.
- Very limited computing power and memory.
- One or more sensors installed in order to make observations.
- A wireless radio capable of transmitting small data packets to a nearby base station and are sometimes also able to receive small packets from the base station.

A network of IoT devices is any group of sensors which collect the same type of data that is not statistically independent of each other. Another important point about mMTC is that there is never or rarely any human in the loop. It is all machine-to-machine communication used to automate processes.

The sensors' observations provide data about the process by sampling it at a specific time in a specific place, i.e. it is tangible and has a predefined datatype. Information is a much broader term; it can be about the process observed, the IoT devices, or anything inferred from data. Sampling is the process of procuring data, i.e., making an observation of a process at a specific time or time interval.

When an IoT device makes a sample, that data needs to be transmitted. In this thesis it is assumed that there is some form of transmission budget. This can be due to

limited battery on the device or to avoid overloading the receiver. Transmission budget comes in two main formats:

- Some form of cost or negative impact for each packet transmitted.
- A maximum number of transmissions allowed, either per device or in total.

It is important to take a transmission budget into account. Without it, the problem of sampling IoT devices becomes less of a problem as the devices can just be sampled all the time.

1.1 Motivation

When taking VoI into account in a communication scheme it can be split into three components that can be considered independently or jointly.

Timely Data needs to be delivered when it is needed.

Relevant Data needs to affect the receiver.

Novel Data needs to provide new information.

The data chosen for transmission has to be Timely, Relevant and Novel. The Timeliness is related to the sub-field Age of Information (AoI) [12]. AoI concerns optimizing the transmission process such that the newest possible sample of a process is available at the receiver. However, there is more to the data being timely. In order to optimize for timeliness, the transmission schedule also has to consider when the data is needed. It is a waste of resources to transmit data that will never be used, if it is known that this is the case before transmission. To optimize for the timeliness beyond the AoI, it is necessary to have some side information about the timing of the use of the data.

Optimizing for the relevance of data being transmitted also depend on side information about how the data will be used. It is common that a summary of a data set will be used for analysis and that individual data points only serve to produce an accurate summary. Side information about which function(s) constitutes the summary for the data can be taken into account when choosing which data points are relevant to transmit, such that only data producing a more accurate summary is transmitted. Another way data might be used is as training data for a neural network. This type of information can be used to optimize the data collection to shorten the training time, by not transmitting data that can already be correctly classified by the network [17].

Novel data is data that cannot be predicted, i.e., the information that the data provide cannot be inferred from other data or predicted using the receiver's estimator. It is not identical to a sample already transmitted by another device. To optimize for novelty the IoT devices can be provided with some rules for what constitutes expected

or unexpected data, or side information about the network can be utilized to predict what can reasonably be inferred and only sample the ones that cannot be predicted.

The difference between the relevance aspect and the novelty aspect can be made clear with the following two examples. Data can be relevant without being novel. An example of this is when constructing a *coreset* of a larger data set, [10]. A *coreset* is a subset of data that approximates the full set, so a sample can be relevant to the *coreset* without being novel. A data point can be novel by being smaller than predicted, but if the only aggregate of interest is the maximum value then that data point is not going to be relevant.

1.2 Thesis Objectives

The main focus of my research has been on utilizing the available side information to optimize the information flow in IoT sensor networks. Such side information can come in many shapes and sizes, so in this work it is limited to some specific categories of side information, and how those can be utilized to ensure that the most useful data is transmitted from the devices in the network.

The first category is information about a graph that can be used to model the relation between the sensors in the network. When the information that the sensors in the network has access to are not independent of each other, a graph can be constructed to describe the dependency. When such a graph is known, it can be used to optimize the sampling and processing of data from the sensors.

The second category is information about when data will be used. Two cases will be considered; firstly, when data from an individual sensor will be used, such that the data collection can be optimized on an individual sensor level to avoid collecting data that will never be used. Secondly, when data from the sensor network as a whole will be used, such that all sensors in the network can be coordinated to prepare for the time when data will be pulled from the network.

The third category is information about which aggregates of the data will be used. In this context, an aggregate is a function that takes some data from the network and outputs one value. Aggregates are often the first step in data analysis or used for decision making. Knowing which aggregates of a data set will be used gives room to optimize the data collection, because not all observations a equally important to determine the correct aggregate for the data set. We do not consider the value of the data collected for the first two categories of side information when choosing how to sample. With this third category we consider the value of data already collected and the expected value of future data to determine which sensor to sample at which times, in order to fully utilize information about the aggregate.

These three categories of side information correspond to the three bases of sampling laid out in the title of this thesis, namely graph-based, time-based, and value-based. The central question of the research is about sampling, i.e. getting the best possible

representation of the information within a limited transmission budget.

- How can knowledge of a graph representing relations between IoT devices be utilized to improve the usefulness of the communication system?
- How can knowledge about the timing of queries for the IoT sensor network be utilized to optimize the response to them?
- How can knowledge about the aggregates be used to summarise the information and optimize the sampling of an IoT sensor network?

1.3 Thesis Outline

The thesis is in two parts. The remainder of part one is organized as follows: Chapter 2 provides theoretical background relevant for the first research question and a summary of contributions for the two papers aimed at that question. Chapter 3 provides theoretical background relevant for the second and third research questions and a summary of contributions for those questions. The background and summary for the second and third question are in the same chapter because there is significant overlap in the background for these. Chapter 4 draws conclusions on each of the three research questions, provides some final remarks and future directions.

The submitted and published papers that constitute the main work of this thesis are included in part two. They are ordered firstly after the research question that they relate to and secondly by date of submission. Paper A and B are aimed at the first question. Paper C, D, and E are aimed at the second question. Paper F and G are aimed at the third question.

2 Utilizing Graph Structure

IoT sensor networks are an important part of the mMTC aspect of 5G. When dealing with such networks, it can be beneficial to represent them as graphs [5, 15], where the nodes represent the IoT devices and the edges represent a relationship between them.

There are many different ways of representing an IoT sensor network with a graph, depending on the purpose of the representation and on what information is available. In cases where information about the geographical placement of the devices is available, it is common to use this information to create the graph. This type of graph can be seen as a naturally induced graph. Examples of this are networks of weather stations [23], sensors observing water distribution networks [29], and sensor in road networks [26]. In the latter two examples, it is not purely geographical information that determines the edges, but rather pipes or roads between devices.

In cases where no such information is available, or the information has proven to be of little use in a particular case, it might be advantageous to make a data induced graph. This requires that there are some data available from the network. In order to get a good fit, a decent amount of data needs to be available from the sensors. The exact amount necessary depends on the method used, of which there are many [16, 24].

When a graph for the IoT devices has been established, a graph signal can be defined. For a network of N devices, $G = \{\mathcal{V}, \mathcal{E}, \mathcal{A}\}$ is the graph, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, and $\mathcal{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. The adjacency matrix representing the N devices is zero on the diagonal and is symmetrical for undirected graphs. $\mathcal{A}_{i,j} > 0$ when there is an edge representing the relation between node i and j. In weighted graphs $\mathcal{A}_{i,j}$ is the weight of the edge, whereas in unweighted graphs all weights are one. The graph signal, defined as $x(v, t)$, is sensor data collected from the network. It is a discrete signal where v is the range of nodes in the graph and t is the range in time.

Much research has been done on processing of signals on graphs [21, 25]. Most of it has been concerned with the simple case of signals without a time dimension. The main definitions and results are extensions from classic signal processing to the irregular, discrete domain of graphs. In the following definitions $x(v)$ will denote a single instance of a graph signal.

The graph Laplacian is $\mathcal{L} = D - \mathcal{A}$, where D is the diagonal matrix with D_i being the number of edges connected to node i. When the graph is undirected, $\mathcal{A}$, and therefore also $\mathcal{L}$, are symmetric. In my work, I only consider undirected graphs and some of the following results does not hold for directed graphs, but can be extended to hold. The Laplacian is a difference operator in the graph:

$$(\mathcal{L}x)(i) = \sum_{j \in \mathcal{N}_i} \mathcal{A}_{ij}(x(i) - x(j)), \tag{1}$$

where $\mathcal{N}_i$ is the set of nodes connected to node i. The graph Fourier transform for a

graph signal x is defined in terms of the eigendecomposition of the graph Laplacian, $\mathcal{L} = U\Lambda U^{-1}$, where the graph Fourier spectrum is sampled in the eigenvalue and the transform is defined as the inner product with the eigenvectors.

$$\hat{x}(\lambda_l) = \langle x, u_l \rangle = \sum_{i=1}^{N} x(i) u_l^*(i). \tag{2}$$

With this definition of the graph Fourier transform, the transform on an unweighted, undirected circle graph reduces to the discrete Fourier transform [21]. The Graph Fourier transform facilitates an interpretation of the frequency of the graph signal. High amplitude of the graph Fourier transform for low eigenvalues mean that the signal changes slowly across the graph. High amplitude for the high eigenvalues mean that the graph signal changes fast [25]. The inverse graph Fourier transform is:

$$x(i) = \sum_{l=0}^{N-1} \hat{x}(\lambda_l) u_l(i). \tag{3}$$

Other classical aspects of signal processing have also been extended to the graph domain, such as filtering in both the frequency- and vertex- domain, convolution, translation, modulation, and dilation. These results are less important in the context of sampling and therefore will not be discuss further here.

Graph signal processing has been used for sampling prior to the work of this thesis. Several works consider sampling of a single time instance of a graph signal under some assumptions about what type of connection between the nodes the graph represents [3, 6, 28]. These papers assume that the signal on the graph is limited in frequency bandwidth. Frequency bandwidth is a measure of how many graph Fourier components are significant in value. It is also assumed that there exists a linear interpolator that can be used to recover the unsampled data.

When collecting data in a network of IoT devices, it is necessary to consider both the graph dimension and the time dimension jointly. When considering a graph-time signal, $x(v, t)$, it is often correlated in both the time- and graph- dimension or even jointly correlated. To optimize the sampling scheme it is crucial to consider both. Some initial work has been done in the field of applying graph signal processing to IoT sensor networks. [29] uses graph signal processing to reduce the number of devices in an IoT network. This can be useful, if the data collection operates on a subscription basis, where a user can save resources by unsubscribing from a device. However, if the users own the devices, the cost of installing the device will be wasted if it is not utilized. [7] proposes a simple importance measure for the devices in the network base on the graph Fourier transform, which is used to sample the graph-time signal.

2.1 Summary of Contributions

In this thesis the following papers on the topic of utilizing graph structure has been written:

Paper A: Josefine Holm, Federico Chiariotti, Morten Nielsen, and Petar Popovski, "Lifetime Maximization of an Internet of Things (IoT) Network based on Graph Signal Processing," *IEEE Communications Letters*, Vol. 25, No. 8, pp. 2763–2767, 2021.

Paper B: Roshni Chakraborty, Josefine Holm, Torben Bach Pedersen, and Petar Popovski, "Finding Representative Sampling Subsets in Sensor Graphs using Time Series Similarities," Submitted to *ACM Transactions on Sensor Networks*, 2022.
Preprint: `https://arxiv.org/pdf/2202.08504.pdf`

These papers explore the use of graph signal processing to optimize the data collection in a IoT sensor network for novelty. The aim is to have the lowest possible error on the reconstruction of unreceived data. When using this type of metric, the data collected is what is considered to hold the most information not already known by the receiver, thus making it an optimization for novelty.

The two papers in this topic share the same model for the wireless data collection. One or more base stations collect data from the devices in the network. The communication between the base station and the devices are wireless and the devices only communicate with the base station. Because the focus of this part of the thesis is on sampling so only the most novel data is transmitted, it is assumed that the error on the transmissions is negligible. Negligible transmission error can be achieved through coding, retransmission, or a combination of the two, often at the cost of some latency or additional transmission cost. With the problems considered in these papers, the communication cost is significantly reduced by sampling, so a small increase in the cost per data point is acceptable. Furthermore, latency is not in focus in these papers because a small delay does not significantly affect the type of reconstruction and analysis considered.

In papers A and B, it is assumed that there exists a relation between the data observed by the sensors in the network, and that it is possible to find a graph representing this relation. A concept drawing of the system model can be see in Figure 1.

In paper A, the graph structure, representing the relation between the data collected by the different devises in the network, is assumed to be naturally induced. The example used for the numerical results is from the water distribution simulation tool EPANET [22], which provides realistic data from a sensor network deployed in water pipes. A graph for the network can be created by drawing an edge between sensors that are connected by a pipe. The objective of the paper is to partition the devices in the network into disjoint groups. These group are then sampled in a round robin manner; the communication can be either pull- or push-based as each device is sampled periodically. The partitioning is optimized for novelty such that each group represents the signal. The main purpose of this is to reduce the energy consumption by the IoT devices,

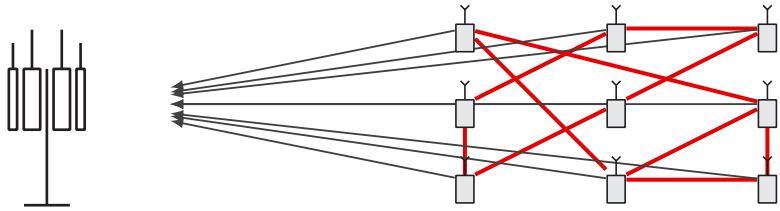


Fig. 1: Depiction of the system showing the communication from the IoT devices to the base station (black arrows) and the relation between the devices (red lines).

as they are often battery powered. Saving battery power extends the lifetime of the device, reducing the need for maintenance, and replacement. Reduction in number of transmissions also decreases the load on the base station and reduces the interference.

Paper B extends paper A in two main ways: Firstly, it extends with a survey of data induced graph creation methods. Secondly, it extends with further research into optimal sampling partitions. The first part of paper B is a survey on graph creation methods that uses historic data from the network to create a graph that is optimal in some sense. The optimality criteria used to create the graph affect the result, and the resulting graphs have different properties that can be useful for different applications. Therefore, there is not a right or wrong method in general, as it depends on the network and the application considered. The survey considers six different methods and compares the properties of the resulting graphs. The second part of paper B continues the investigation of the problem of partitioning the IoT devices into disjoint sampling sets in an optimal manner, as proposed in paper A. In paper A, a single measure of optimality is utilized in a novel algorithm for creating the optimal sampling scheme. In paper B the research is expanded to include four additional measures of optimality, which are used to solve the problem of creating the highest number of disjoint sampling sets given an error bound and its dual problem. This results in six novel algorithms. The graph creation methods and sampling scheme algorithms are tested on a variety of simulated and real world data sets of different types, such as recordings of pressure, temperature, and humidity. The tests consider all combinations of graph creation method and sampling scheme algorithm to find the best combinations for the different types and sizes of data sets. Based on initial test, recommendations are formed depending on network size and desired edge density of the graph, these are confirmed though a second round of tests.

3 Age of Information and Beyond

Timing and timelines have received significant attention with the rise of 5G technology, [18]. For a wireless transmission to be successful, there is a series of actions that needs to be performed. These include the encoding, decoding, and the transmission itself. All of these take time and have strict requirements in order to be considered real-time. Such requirements are the latency budget for the transmission [2]. The relevant latency budget for an IoT device depends on the hardware and software of that device, as well as on the application for which it is used. The notion of latency for real-time applications is considered one of the main features of 5G wireless systems.

Age of Information (AoI) is a timing metric related to latency, that describes the age of the most recent packet received at the destination. Age is the time since the packet was generated, and in the case of a device with one or more sensors it is the time since the sensor made the measurement. The AoI is thus dependent on two factors: the packet generation time and the transmission latency. AoI is most often a linear function measuring the freshness. However, in some scenarios it can provide valuable insight to use a nonlinear AoI function [27]. In the remainder of the thesis, it is assumed that the AoI is a linear function, and other cases will not be discussed further as they have limited relevance to the contributions of this thesis. AoI assesses sampling and transmission collectively rather than separately.

Traditionally many systems used a push-based communications model, where a packet was created and transmitted whenever something new happened. With the derivatives of AoI and VoI considered in this work, the communication model is pull-based, where the receiver makes a request for an update when needed. This is inspired by the field of databases in computer science, where the users make formal requests to the database, known as queries. The type of query considered here can ask for aggregates or subsets of data from the Data Base Management System (DBMS), [9, Section 1.1]. The task of the computer scientist is to optimize the system such that queries are responded to as efficient as possible. Similarly, in a pull-based communication model, data is requested from devices when needed.

Examples of important derivatives of AoI include Peak Age of Information (PAoI), which measures the maximum AoI before the receiver gets an update from the transmitter [11]. Also, [13] considers a case where multiple servers receives infrequent and asynchronized updates about the same process and the Server have different response time. A user can send identical requests for data from multiple servers. The paper finds that waiting for multiple responses from the server reduces the AoI on average.

The next level after considering AoI and its derivatives and optimizing the timing, is to start considering the content of the data itself [19]. An example of how this can be done in a push-based communication model, where the transmitting side is given a set of rules or criteria to decide if the data it has available will be important to the receiver. In most cases, the receiver has a model for the data and the goal is to transmit

updates whenever the model is wrong. If the transmitter has the necessary capacity, it can have a copy of the model and push updates when appropriate. However, this is rarely the case. One way to work around this is for the receiver to predict when the model is insufficient and pull data to correct it. Taking the content of data into account when optimizing the sampling scheme is often referred to as Value of Information (VoI) or goal-oriented communications [4].

There are different ways of measuring the importance or value of information. In [1] the authors consider a case where the importance of observed events are tied to the frequency of the event occurring, so less frequent events are considered more important to report than more frequent events. Other examples of derivatives of VoI are Age of Incorrect Information (AoII), [14], which extends AoI by not only considering fresh updates but fresh "informative" updates. Informative here is defined as reducing the estimate that the receiver have of the process observed by the transmitter. Secondly, Urgency of Information (UoI), [30], is an extension of VoI that specifically aim to stabilize the remote control of an actuator. In this case, information that affect the control decisions should be prioritized.

3.1 Summary of Contributions

In this thesis, the following papers on the topic of utilizing graph structure has been written:

Paper C: Josefine Holm, Anders E. Kalør, Federico Chiariotti, Beatriz Soret, Søren K. Jensen, Torben B. Pedersen, and Petar Popovski, "Freshness on Demand: Optimizing Age of Information for the Query Process," *IEEE International Conference on Communications*, pp. 1–6, 2021.

Paper D: Federico Chiariotti, Josefine Holm, Anders E. Kalør, Beatriz Soret, Søren K. Jensen, Torben B. Pedersen, and Petar Popovski, "Query Age of Information: Freshness in Pull-Based Communication," *IEEE Transactions on Communications*, vol. 70, no. 3, pp. 1606–1622, 2022.

Paper E: Søren Kejser Jensen, Josefine Holm, Federico Chiariotti, Christian Thomsen, Anders Ellersgaard Kalør, Petar Popovski, Beatriz Soret and Torben Bach Pedersen, "SENDAI: A Framework for Joint Reasoning About Sensor Data Acquisition and Sensor Data Analytics," Submitted to *Information and Computation Journal*, 2022.

Paper F: Federico Chiariotti, Anders E. Kalør, Josefine Holm, Beatriz Soret and Petar Popovski, "Scheduling of Sensor Transmissions Based on Value of Information for Summary Statistics," *IEEE Networking Letters*, vol. 4, no. 2, pp. 92–96, 2022.

Paper G: Josefine Holm, Federico Chiariotti, Anders E. Kalør, Beatriz Soret and Petar Popovski, "Goal-Oriented Scheduling in Sensor Networks with Application Timing Awareness," submitted to *IEEE Transactions on Communications*, 2022.

Paper C defines a new derivative of AoI named Age of Information at Query (QAoI). While AoI is a continues measurement, and optimization ensures that the age is low at all times, Age of Information at Query (QAoI) instead measures discretely at the time when a query arrives at the receiver. The aim of this is to promote having fresh information when it is needed for a query and avoid wasting resources on updates that will never be used. To optimize for QAoI each device in the network is considered individually and sampled using a pull-based communication model. The receiver has information about the query and pulls the device for updates when it is deemed appropriate. The transmission budget is modeled as a leaky bucket, where tokens are generated in a random process and the device can only transmit if it has a token available. The receiver knows the token status. The communication channel is modeled as a packet erasure channel with constant error probability for simplicity.

The results of optimizing for this using a Markov decision process shows that it is possible to have lower age at the query time by saving up tokens and rapidly using them immediately before a query. When comparing with the traditional sawtooth pattern of an AoI optimized scheme, there is much more variety in the size of the peaks. The peaks are much taller in the middle of a query interval and very small right before a query. When the exact time of the query is known, this method is effective at minimizing the age. However, the results also show that if the query arrival process is stochastic, and

the only information available is statistical, the possibility for foresighted scheduling can be diminished.

Paper D is an extension of paper C. It provides further in depth analysis of the QAoI measure. More channel models are considered so that more factors are taken into account when optimizing. Paper D also provides a comparison with other AoI derivatives.

Paper E combines sensor data acquisition with storage and data analysis. It proposes Framework for joint Sensory Data Acquisition and Analysis (SENDAI) as a tool to optimize these different phases in the life of data. The framework is based on two very disparate real-life use cases, provided by industry collaborators. It provides a generalized way of describing and optimizing the journey of data, from the state of the process to the end user. When the data acquisition and analysis are considered jointly, it enables the system to use information about how and when the data will be used to optimize the sampling scheme, storage and analysis, as well as taking into account the limitations of each layer when optimizing the others.

Paper F and G defines VoI in terms of aggregation functions, such that the measure can be tailored to the specific user or application. The two papers consider a similar system model with multiple IoT devices with time-slotted wireless links to a base station. The base station has storage and computation resources and can pull data from any device at any time slot. The devices each observe one part of a multidimensional process. The base station keeps a model of the process and computes aggregates that are sent to users when requested.

Paper F considers the case where a request arrives a the base station and the base station then have time to pull one device before responding. This is the one-step optimal solution where, given some historic data, the aim is to determine which device improve the confidence in the response the most. Four different aggregation functions are considered: sample mean, sample variance, maximum, and number of state components within an interval.

Paper G extends the work of paper F in two ways. Firstly, the time aspect is extended, so instead of having one time-step to optimize for a query, the type and timing of the queries are known further ahead of time so multiple devices can be pulled to optimize for the query. Secondly, the case where multiple users makes requests for different aggregates at different times is considered. This means that the scheduler at the base station have to optimize for different functions of the data at different time and balance the aggregates in the optimization. These two extensions significantly increase the complexity of the problem. To this end, the paper proposes a deep reinforcement learning scheme able to predict upcoming queries and pull the devices such that it outperforms naive approaches.

4 Conclusion

How can knowledge of a graph representing relations between IoT devices be utilized to improve the usefulness of the communication system?

Paper A and B proposes methods for energy-efficient sampling of IoT devices. The main focus is on partitioning the devices into disjoint sampling sets using a graph structure for the devices. The purpose of the partitioning is to lower the communication load on the system, in order to save energy of the IoT device side and reduce the risk of collision and interference on the base station side. The partition is optimized such that the reconstruction of the observed process is as good as possible at all times. This is used as a measure of novelty, as the set that results in the lowest reconstruction error is the one best suited for interpolating the rest of the signal.

Two types of graphs are used to represent the relations between the IoT devices; naturally induced graph based on side information about the system and data induced graph based on historic data from the sensors on the devices. The results show that using a measure of importance on the graph, that takes into account all the devices in a set jointly, results in lower reconstruction error than assessing them individually or assigning them randomly.

How can knowledge about the timing of queries for the IoT sensor network be utilized to optimize the response to them?

Paper C and D proposes a pull-based sampling scheme for optimizing the response time for incoming queries. The new metric for measuring the age of information at the time it is needed for a query has been dubbed QAoI. Optimizing for this metric produces a sampling scheme that is significantly different from one optimized for classic AoI. The papers show that the new metric and proposed scheme leads to good results when the query process is well known. This works even in cases with spotty connection, such as satellite communication, where the channel is only open for short periods.

Paper E further contributes to this research with a discussion on the trade-of between push- and pull-based communication. Either push, pull, or a combination can be used depending on the information available and the storage and computational capacity of the different parts in the system.

How can knowledge about the aggregates be used to summarise the information and optimize the sampling of an IoT sensor network?

Paper F and G proposes policies for sampling a network of IoT devices optimized for relevance. This is done for multiple different aggregation functions. Paper F derives the one-step optimal solution and shows that the optimal device to pull is significantly different for the considered aggregates, as for some of them the error compounds and for other aggregates such that they compensate for each other.

Paper G uses reinforcement learning to solve a more complex version of the problem. Multiple different aggregates of the same process are considered, as well as their long-term impact. This leads to better policies than one-step greedy strategies.

4.1 Final Remarks and Future Directions

In this work, the three aspects of Value of Information; Timeliness, Relevance and Novelty, have been discussed and optimized for, both separately and jointly. Based on this, it can be concluded that including multiple aspects increases the complexity of the optimization problem, but making use of all available side information also leads to better solutions.

Several avenues are open for future work, such as increasing the complexity of the problems considered and combining the proposed methods. Firstly, the problem could be extended to include more complex data types, such as point clouds, images, or multivariate data, i.e. IoT devices with multiple different sensors from which updates can be transmitted as a single packet. This will increase the complexity of the prediction and thus requires estimators capable of dealing with more complex functions, data types, and system models. Secondly, different methods proposed in this thesis could be combined to utilize the knowledge of a graph structure for the IoT devices, in order to increase the accuracy of the prediction. Alternatively, the proposed graph-based sampling algorithms can be used to decrease the energy consumption in the system, while optimizing for timing, relevance, or novelty.

References

[1] P. Agheli, N. Pappas, and M. Kountouris, "Semantics-aware source coding in status update systems," in *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2022, pp. 169–174.

[2] Y. Alfadhli, Y.-W. Chen, S. Liu, S. Shen, S. Yao, D. Guidotti, S. Mitani, and G.-K. Chang, "Latency performance analysis of low layers function split for urllc applications in 5g networks," *Computer Networks*, vol. 162, p. 106865, 2019.

[3] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3864–3868.

[4] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 109–117.

[5] O. M. Bushnaq, A. Chaaban, S. P. Chepuri, G. Leus, and T. Y. Al-Naffouri, "Sensor placement and resource allocation for energy harvesting iot networks," *Digital Signal Processing*, vol. 105, p. 102659, 2020.

[6] L. F. Chamon and A. Ribeiro, "Greedy sampling of graph signals," *IEEE Transactions on Signal Processing*, vol. 66, no. 1, pp. 34–47, 2017.

[7] A. Chiumento, N. Marchetti, and I. Macaluso, "Energy efficient wsn: a cross-layer graph signal processing solution to information redundancy," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2019, pp. 645–650.

[8] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.

[9] R. Elmasri, S. B. Navathe, R. Elmasri, and S. Navathe, *Fundamentals of Database Systems*, 7th ed. Springer, 2000.

[10] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011, pp. 569–578.

[11] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *2015 IEEE international symposium on information theory (ISIT)*. IEEE, 2015, pp. 1681–1685.

[12] 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.

[13] F. Li, Y. Sang, Z. Liu, B. Li, H. Wu, and B. Ji, "Waiting but not aging: Optimizing information freshness under the pull model," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 465–478, 2020.

[14] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides, "The age of incorrect information: A new performance metric for status updates," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2215–2228, Jul. 2020.

[15] M. Mangia, F. Pareschi, R. Varma, R. Rovatti, J. Kovačević, and G. Setti, "Rakeness-based compressed sensing of multiple-graph signals for iot applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 682–686, 2018.

[16] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[17] I. Nikoloska, J. Holm, A. E. Kalør, P. Popovski, and N. Zlatanov, "Inference over wireless iot links with importance-filtered updates," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1089–1098, 2021.

[18] P. Popovski, F. Chiariotti, K. Huang, A. E. Kalør, M. Kountouris, N. Pappas, and B. Soret, "A perspective on time toward wireless 6g," *Proceedings of the IEEE*, vol. 110, no. 8, pp. 1116–1146, 2022.

[19] P. Popovski, O. Simeone, F. Boccardi, D. Gündüz, and O. Sahin, "Semantic-effectiveness filtering and control for post-5g wireless connectivity," *Journal of the Indian Institute of Science*, vol. 100, pp. 435–443, 2020.

[20] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view," *Ieee Access*, vol. 6, pp. 55 765–55 779, 2018.

[21] G. B. Ribeiro and J. B. Lima, "Graph signal processing in a nutshell," *Journal of Communication and Information Systems*, vol. 33, no. 1, 2018.

[22] L. A. Rossman *et al.*, "EPANET 2: users manual," 2000.

[23] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[24] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, 2017.

[25] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging filed of signal processing on graphs," *IEEE Signal Processing Magazine*, 2013.

[26] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "A windowed graph fourier transform," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*. Ieee, 2012, pp. 133–136.

[27] Y. Sun and B. Cyr, "Sampling for data freshness optimization: Non-linear age functions," *Journal of Communications and Networks*, vol. 21, no. 3, pp. 204–219, 2019.

[28] F. Wang, Y. Wang, and G. Cheung, "A-optimal sampling and robust reconstruction for graph signals via truncated neumann series," *IEEE Signal Processing Letters*, vol. 25, no. 5, pp. 680–684, 2018.

[29] Z. Wei, A. Pagani, G. Fu, I. Guymer, W. Chen, J. McCann, and W. Guo, "Optimal sampling of water distribution network dynamics using graph Fourier transform," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1570–1582, 2019.

[30] X. Zheng, S. Zhou, and Z. Niu, "Urgency of information for context-aware timely status updates in remote control systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7237–7250, 2020.

Part II

Papers

Paper A

Lifetime Maximization of an Internet of Things (IoT) Network based on Graph Signal Processing

Josefine Holm, Federico Chiariotti, Morten Nielsen, and Petar Popovski.

The paper has been published in the
IEEE Communications Letters Vol. 25, No. 8, pp. 2763–2767, 2021.

The layout has been revised.

Abstract

The lifetime of an Internet of Things (IoT) system consisting of battery-powered devices can be increased by minimizing the number of transmissions per device while not excessively deteriorating the correctness of the overall IoT monitoring. We propose a graph signal processing based algorithm for partitioning the sensor nodes into disjoint sampling sets. The sets can be sampled on a round-robin basis and each one contains enough information to reconstruct the entire signal within an acceptable error bound. Simulations on different models of graphs, based on graph theory and on real-world applications, show that our proposal consistently outperforms state-of-the-art sampling schemes, with no additional computational burden.

1 Introduction

The expansion of Internet of Things (IoT) systems leads to massive data produced from a vast variety of connected devices and sensors, providing unprecedented knowledge about the state and processes of the physical world [1] [2]. These IoT sensors are often cheap, wireless and battery-powered. They transmit data sporadically to a Base Station (BS), which forwards the data to a data analytics module. In order to increase the longevity of the massive IoT network, each sensor should try to minimize the number of transmissions, while not compromising the quality of the inference at the data analytics module. This letter addresses the optimization of the conflicting objectives of network lifetime and correctness of inference by casting the problem in the context of signal processing on graphs.

We consider a deployment of an IoT network in which the underlying graph structure is induced naturally by the physical deployment of the IoT devices. This concept is shown on Figure 1, where the IoT sensors are attached at the vertices of a water supply network. Other similar examples include IoT devices attached to other types of utility networks, sensors deployed along streets, etc. Each sensor transmits the data to a BS. Due to the physical setup, the readings of the sensors connected within the physical graph are correlated and sampling of one of them carries information about the potential reading of the other sensor. The objective is to partition the set of sensors into S subsets, such that the subsets are disjoint and all sensors are assigned to exactly one set. In each *sampling round* the data analytics module reconstructs the state of the graph based only on the transmission of a subset of sensors. This reduces the duty cycle requirements for each sensor, as it has to transmit S times less often. Partition sampling can then, in principle, increase the IoT network lifetime by S times, which can be equivalent to an order of magnitude increase. However, it comes at the cost of increased error in the reconstructed signal at the receiver, as the data are incomplete. Finding a partition that minimizes that error is critical to achieve the best possible balance between increased

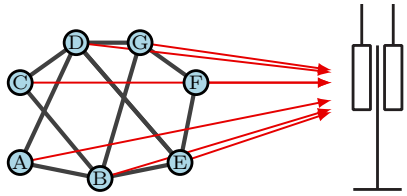


Fig. A.1: Concept drawing of the system on a slice of the test network. Communication between base station and sensors (red arrows) and associations between sensors (black lines).

battery lifetime and reconstructed signal quality.

The reconstruction of the state can leverage the methods of the emerging field of signal processing on graphs [3, 4]: the graph structure, representing the correlation among the measurements from the different sensors, can be exploited to improve the reconstruction of the signal from each of these subsets. The optimal partition of the graph, i.e., the one that provides the lowest reconstruction Mean Square Error (MSE) for a given number of subsets S, is a combinatorial problem, as the value of adding a node to the sampling set depends on the other nodes already in the set. Most works in the literature have concentrated on finding a single set, and not a complete partition. To the best of our knowledge, the first work to do so was [5], which considered band-limited signals, and find a sampling set that allow for perfect reconstruction under certain conditions. A later work [6] aimed at finding the smallest possible set for a given MSE bound, while [7] concentrated on the opposite problem, i.e., finding the sampling set of fixed size with the minimum MSE. A modification of [6] that is resilient to packet losses was presented in [8], but the authors still do not take into account the fact that the data stemming from the graph nodes are a time series, while [9] considers the temporal aspect but neglects to consider the fact that the importance of a node depends on the other nodes in the subset.

The possibility of exploiting graph structures and correlations in the data from different IoT sensors has already been considered for water distribution networks [10], which are uniquely suited for this thanks to the strong correlation between nodes and lack of high-frequency dynamics in the graph. In general, reducing the activity of each sensor can give huge benefits in IoT networks, as sensors are generally battery-powered and expected to last for several years.

This work extends the heuristic from [6] towards finding the largest complete partition, all subsets of which respect the MSE constraint. Our main contribution is a new algorithm that allows us to find the largest partition of the set of nodes in the graph that respects an MSE constraint. While our algorithm is a heuristic, it can be very close to the optimal by considering the effect of the existing nodes in each subset

when adding nodes. This work shares some similarities with [9], but we take the correct importance of each node, considering the elements already in the various subsets, instead of simplifying the problem by assuming its independence from other elements. Our algorithm can run with a similar computational cost to [6] even though we find a complete partitioning, and reduces the average MSE by approximately 5%.

2 System Model

Consider a network of IoT devices wirelessly transmitting to a Base Station. Such a network can be described as a graph where the sensors are represented with nodes and the edges represent similarity between the sensors in terms of what values they observe. The edges can be organized into an adjacency matrix $A \in \mathbb{R}^{N\times N}$, where N is the number of nodes and $A_{i,j} \neq 0$ if and only if there is an edge between node i and j. The observations of the sensors are considered a signal, this signal exist in both time and space, such that for each sensor there is a time series and for each timestamp there is a graph signal. We will denote the graph signals by $x \in \mathbb{R}^N$. In this letter we propose a method of prolonging the lifespan of the sensors. We split the sensors in disjoint sets and, at a given timestamp, the Base Station (BS) samples only one set. If the sets are sampled on a round-robin basis, then the channel usage is reduced and thus the network lifetime is prolonged. The problem is how to do this and still be able to reconstruct the full graph signal at each timestamp within a reasonable margin of error.

In order to solve this problem we utilize concepts from graph signal processing [3]. The graph Fourier transform is defined in terms of the of the eigenvectors of the graph Laplacian defined as $L = D - A$, where D is the degree matrix, i.e. a diagonal matrix whose i-th element is the sum of the weights going into vertex i. We write the eigen decomposition as $L = VEV^H$, where $V = [v_1 \cdots v_N]^H$ is a matrix with the eigenvectors of L and E is a diagonal matrix with the corresponding eigenvalues. We use the eigen decomposition to define the Fourier transform of x on the graph as $\bar{x} = V^H x$.

If we assume that $x \in \mathbb{R}^N$ is a spectrally sparse signal on the graph, i.e., $x = V_{\mathcal{K}}\bar{x}_{\mathcal{K}}$, where $|\mathcal{K}| \ll N$ and $V_{\mathcal{K}} \in \mathbb{R}^{N\times|\mathcal{K}|}$ are the columns of V with index in the set $\mathcal{K}$, then compression is obtained in the graph Fourier domain. Each node i can measure one component of the signal, with an additional noise component: the fully sampled signal is then $y = x + w$, where the noise, w, follows a zero mean circular distribution with $\Lambda_w = \mathrm{diag}(\lambda_{w,i})$. We remark that this noise is inherent in the measurement operation, and is not due to the communication link with the BS. It is further assumed that the noise has full rank. Even if the sampling set contains every node in the graph, the reconstruction of the signal at the BS will never be perfect due to this measurement noise. If we consider sampling sets that are strict subsets of the graph, the reconstruction quality will degrade accordingly.

This imperfect reconstruction from smaller sampling sets is the one we consider: as in [9], the goal is to divide the sensors into disjoint sampling sets, where each set produce

good enough representation of the entire graph signal on its own. The approach in [9] is to calculate the importance of of each node individually and then use that to group the nodes. However, that approach does not take into account that the importance of the nodes depends on which nodes they are grouped with. This is the main difference from our approach, as we recalculate the importance at each step to take maximum advantage of the graph structure. To this end, we use the reconstruction MSE for a sampling set as in [6]. The authors in [6] define the optimal linear interpolation operator to obtain the reconstructed signal $\hat{x} = By$, accounting for the fact that the importance of the nodes depends on the other nodes in the set; however, the objective is to construct only one sampling set. Specifically, the goal in [6] is to make the smallest possible sampling set with reconstruction error below a given threshold. An optimal linear interpolation operator B^* can be found for each sampling set by minimizing the interpolation error covariance matrix

$$B^* = \arg\min_B \quad \mathbb{E}[(x - By)(x - By)^H | x, w]. \tag{A.1}$$

In the following, we use the simplified notation $B^*y = \hat{x}$ to indicate the reconstructed vector after interpolation. Using the approach from [6] as a starting point and putting it in the context of network longevity, we create several disjoint sampling sets that are all sufficiently good to reconstruct the original graph signal below the error tolerance. We will denote the partitioning G, G_p is set number p and $G_{p,j}$ is iteration j of set p. It is defined as $\text{MSE}(\hat{x}) = E\|x - \hat{x}\|_2^2$. As we follow the same model as defined in [6], we can exploit their derivation of the maximum MSE for the sampling set:

$$\text{MSE}(G_p) = Tr[K(G_p)], \tag{A.2}$$

where

$$K(G_p) = V_{\mathcal{K}} \left(\Lambda^{-1} + \sum_{i \in G_p} \lambda_{w,i}^{-1} v_i v_i^H \right)^{-1} V_{\mathcal{K}}^H. \tag{A.3}$$

In [6, Prop 1] it is shown that (A.3) is the error covariance matrix of $\hat{x}$ if we use the optimal interpolation $\hat{x} = B^*y$ from (A.1) given a sampling set. This can be rewritten to calculate the MSE iteratively as follows:

$$Tr[K(G_p)] = V_{\mathcal{K}}^H V_{\mathcal{K}} \left(\Lambda^{-1} + \sum_{i \in G_p} \lambda_{w,i}^{-1} v_i v_i^H \right)^{-1}, \tag{A.4}$$

because trace is rotational, i.e., $Tr[ABCD] = Tr[CDAB]$. We then define:

$$A = \Lambda^{-1} + \sum_{i \in G_p} \lambda_{w,i}^{-1} v_i v_i^H. \tag{A.5}$$

Let K_j be a shorthand for $K(G_p)$ where G_p has j nodes. We will use this for the iterative calculation of $K(G_p)$.

$$Tr[K_{j-1}] = Tr[V_{\mathcal{K}}^H V_{\mathcal{K}} A^{-1}] \tag{A.6}$$

$$Tr[K_j] = Tr[V_{\mathcal{K}}^H V_{\mathcal{K}} (A + \lambda_{w,s}^{-1} v_s v_s^H)^{-1}]. \tag{A.7}$$

We then use the matrix inversion lemma

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1}, \tag{A.8}$$

using $A = A$, $U = v_s$, $V = v_s^H$ and $C = \lambda_{w,s}^{-1}$ we get

$$Tr[K_j] = Tr[K_{j-1} - V_{\mathcal{K}}^H V_{\mathcal{K}} \frac{A^{-1} v_s v_s^H A^{-1}}{(\lambda_{w,s}^{-1})^{-1} + v_s^H A^{-1} v_s}]. \tag{A.9}$$

Therefore, the iterative calculation of of the MSE becomes:

$$\text{MSE}(G_{p,j} \cup v_s) = Tr[K_j] - \frac{v_s^H K_j V_{\mathcal{K}}^H V_{\mathcal{K}} K_j v_s}{\lambda_{w,s} + v_s^H K_j v_s}, \tag{A.10}$$

where

$$K_j = K_{j-1} - V_{\mathcal{K}}^H V_{\mathcal{K}} \frac{K_{j-1} v_u v_u^H K_{j-1}}{\lambda_{w,u} + v_u^H K_{j-1} v_u}, \tag{A.11}$$

$K_0 = \Lambda$ and u is the index for the most recently added node. Note that $V_{\mathcal{K}}^H V_{\mathcal{K}} = I$ for $|\mathcal{K}| = N$, as well as:

$$\text{MSE}(v_s) = \text{MSE}(\emptyset \cup v_s). \tag{A.12}$$

The main problem we will propose a solution to is formulated as:

$$\begin{aligned} &\text{maximize} && k \\ &\text{subject to} && \text{MSE}(G_p) \leq \varepsilon \quad \forall\, p = 0, ..., k. \end{aligned} \tag{A.13}$$

3 Algorithm

The problem in (A.13) is combinatorial, as the MSE of each subset depends on all the elements of the set, and cannot be solved efficiently. For this reason, we propose a heuristic Joint Iterative Partitioning (JIP) algorithm which builds on the basic idea of the Individual Iterative Partitioning (IIP) algorithm from [9, Algorithm 1]. However, the importance of each node is not considered to be fixed, but we compute the possible gain from adding it to a sampling set considering the nodes already in the set: in this way, we can improve the balance between the sets by putting each node in the subset where it can improve the MSE the most.

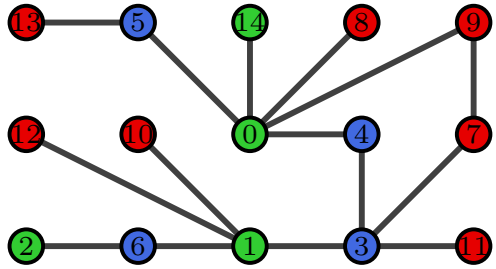


Fig. A.2: Graph partitioned into 3 sets.

The idea behind JIP is to start by calculating the MSE for all possible sampling sets with one node and sorting the nodes based on this. We take the largest node and add it to the first set. If MSE $< \varepsilon$ the process is done, otherwise we search for the best node to add. If more than one node makes MSE $< \varepsilon$ we choose the one with the smallest MSE alone and the set is finished. If no such node exists, we choose the one that minimizes the MSE for the set. Then we find the next node to add to the set. An example of how this partition is made can be seen in Fig. A.2. The graph is divided into three sets by JIP, which iteratively adds the nodes to the sets. The numbers on the nodes in the figure indicate the order in which they were added, while their color indicates the set in which they were placed. The first node to be assigned is the one with the most neighbors, which makes sense, as its low distance from several others means that the reconstruction drawn only from that node has the lowest MSE. The second node added covers a part of the graph that is far from the first node, but it is still connected to several other nodes. After the third node, the green set is over the threshold, so the algorithm moves to a new set. As the best nodes are selected first, the number of nodes in the sets increases, and the red set has 7 nodes. The last node is allocated to a random set, as it cannot form a set on its own and all existing sets are below the threshold MSE. The allocation of these "orphan" nodes, which cannot form another set with the required MSE, can be performed in various ways, with the effect of further reducing the MSE, but it cannot increase the number of sets in the partition.

The algorithm requires $O(n^2|\mathcal{K}|^2)$ operations, the same as [6] in the worst case scenario. The pseudo-code for our approach is given in Algorithm 1.

Ones the sampling sets are selected we create the sampling matrix S by setting $S_{i,j}$ as 1 if $i \in G_j$ and 0 otherwise, and then get the samples $y_j = S_j x$.

Adaptation of IIP for comparison: In this section we adapt the IIP algorithm from [9] to this setting. The setting is similar in that we also want to minimize the energy consumption of the network by splitting the nodes into disjoint sampling sets. The Authors in [9] assumes that data from the sensors are available and use this to infer a graph structure. In this paper we assume that the graph structure is given,

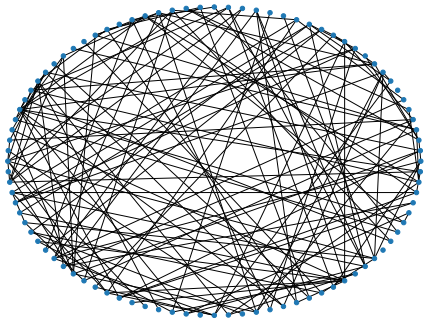
Algorithm 1 Joint Iterative Partitioning (JIP)

1: Input: $\Lambda, \Lambda_w, V, \varepsilon$
2: $K_0 = \Lambda, p = 0, j = 1$
3: $L_s = \mathrm{MSE}(v_s)$
4: $L\_index = argsort(L)$ (largest first)
5: **while** $L\_index \neq \emptyset$ **do**
6: $u = pop(L\_index_{-1})$
7: $G_{p,j} = u$
8: **while** $\mathrm{MSE}(G_{p,j}) > \varepsilon$ and $L\_index \neq \emptyset$ **do**
9: Calculate K_j according to (A.11)
10: **for** i in L_index **do**
11: **if** $\mathrm{MSE}(G_{p,j} \cup L_i) < \varepsilon$ **then**
12: $G_{p,j+1} = \{G_{p,j} \cup L_i\}$
13: $j = 0, p = p + 1, delete(L\_index = i)$
14: goto 5
15: $G_{p,j+1} = \{G_{p,j} \cup \arg\min(\mathrm{MSE}(L))\}$
16: remove chosen node from L_index, $j = j + 1$
17: **if** $\mathrm{MSE}(G_{-1,-1}) > \varepsilon$ **then**
18: Split the nodes in G_{-1} among the other sets
19: Return G

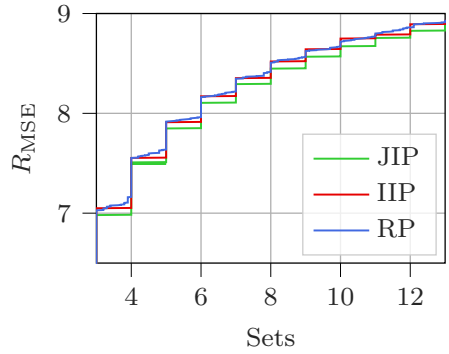
but we have no assumption about the availability of data from the nodes. Therefore, we will skip step 2 and 3 in IIP and instead add the known graph structure to the initialization step. Once a graph structure has been established, the algorithm orders the nodes in descending order of sampling importance in steps 4 and 5. We make a slight modification to this step, as the original algorithm used a Root MSE (RMSE) measure to sort the nodes, while we take the expected MSE which can be computed by creating the sampling matrix for the known graph structure. Once the nodes are sorted, they are iteratively assigned to sampling sets following a greedy approach: the first nodes, i.e., the ones with the lowest RMSE, are put in the first set, until the RMSE of the reconstruction is below the threshold, in steps 10 and 11. As we use the expected value of the RMSE instead of the statistical average, we cannot preform a reconstruction to perform this check, so we just check if the expected RMSE is below the threshold. With this adaptation, IIP from [9] can be compared fairly to the proposed algorithm.

4 Numerical Results

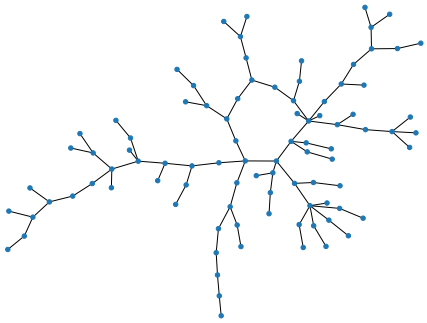
Figure A.3b shows the results of three partitioning methods used on the ER graph structure. It shows a small gain from using JIP rather than IIP or Random Partitioning



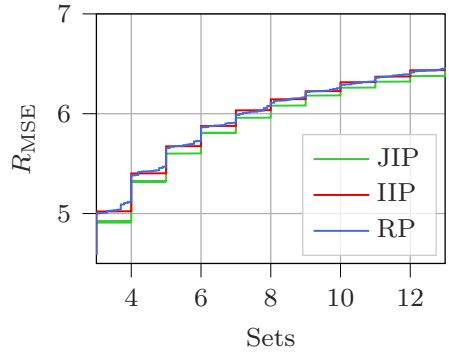
(a) Graph of the Erdős–Rényi (ER) network.



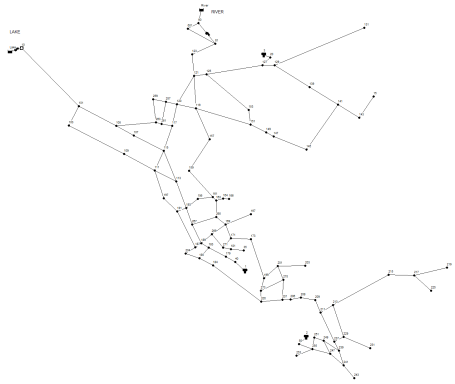
(b) Relative error for different number of sets created for the ER graph structure.



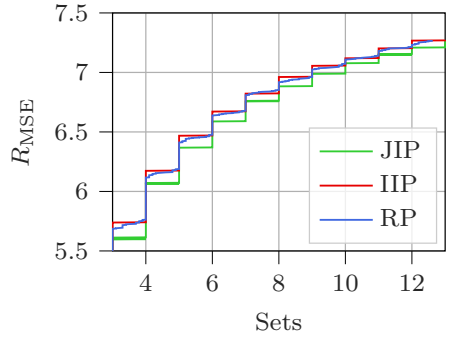
(c) Graph of the Watts–Strogatz (WS) network.



(d) Relative error for different number of sets created for the WS graph structure.



(e) Graph of the EPANET network.



(f) Relative error for different number of sets created for the EPANET graph structure.

Fig. A.3: The graphs and the results.

Algorithm 2 Random Partitioning (RP)

Input: $\Lambda, \Lambda_w, V, \varepsilon$
$K_0 = \Lambda, p = 0, j = 1$
$L\_index = [0, 1, ..., N]$
while $L\_index \neq \emptyset$ **do**
    $r = random(0, |L\_index|)$
    $u = pop(L\_index_r)$
    $G_{p,j} = u$
    while MSE$(G_p) > \varepsilon$ and $L\_index \neq \emptyset$ **do**
        Calculate K_j according to (A.11)
        $r = random(0, |L\_index|)$
        $u = pop(L\_index_r)$
        $G_{p,j+1} = \{G_{p,j} \cup u\}$
        if MSE$(G_{p,j+1}) < \varepsilon$ **then**
            $j = 0, p = p + 1$
            goto 4
        $j = j + 1$
if MSE$(G_{-1,-1}) > \varepsilon$ **then**
    Split the nodes in G_{-1} among the other sets
Return G

(RP). However, for the method to work well we need to have some structure to the graph. The advantage of the JIP method over simple IIP is that it considers the relations between the nodes already in a sampling set and the new node to be added, so the difference between the methods is expected to be starker for graphs with highly local structures, i.e., more clustered ones. Figure A.3d shows the results for a WS graph structure with $\beta = 0.5$. As expected the gain from using JIP is mush bigger and the relative error is lower for all partitioning methods.

In this section we compare the proposed JIP algorithm to the IIP algorithm and a RP algorithm. The RP algorithm creates one set at a time by adding randomly chosen nodes to the set until it is below the given threshold, the pseudo-code for the RP algorithm is given in Algorithm 2. We will compare the three methods on three different graph structures, namely, an ER graph, a WS graph and a graph simulated with EPANET, [11], a tool made for simulation of water supply networks, as shown in Figure A.3e. The EPANET graph can represent a real-world application of the use of our graph sampling technique in IoT networks, although it is still simulated. All graphs are undirected, unweighted and has a similar edge density.

The sampling noise λ_w is included in the calculation of the MSE, therefore we want to present the results of the different algorithms and graphs relative to the MSE sampling

set that is only affected by noise i.e. the set with all nodes, F: $R_{\text{MSE}}(G) = \frac{\text{MSE}(G)}{\text{MSE}(F)}$.

Lastly, Figure A.3f shows the results for the graph simulated with the EPANET tool. This graph has some clustering, but not as mush as the WS graph, and the results show that both the relative error and the gain form using JIP is somewhere in between the results for ER and WS, which is what we can expect from most real world graphs.

In order to show the optimality gap of our heuristic, we have performed an Exhaustive Search (ES) for all possible partitions on two graphs with 15 nodes, a WS graph and an ER graph. Performing ES on larger graphs is infeasible, because the computation time grows exponentially. Figure A.4 shows the MSE of the best possible partitioning with two, three and four sets, measured by the maximum MSE for any set in the partitioning. For the three greedy sampling algorithms it shows the lowest possible threshold that gives the same amount of sets. As the figure shows, JIP is significantly closer to the optimum than IIP or RP in most cases, particularly on the WS graph structure. For the ER graph all sampling algorithms struggled with the partition in two sets, but for three and four sets we lower the MSE threshold by 30-40% compared to IIP and 40-50% compared to RP. For a theoretical analysis of the optimality of the proposed algorithm we refer to [6]. They perform an in-depth analysis of the approximate supermodularity of using (A.2) for greedy optimization.

The main advantage of our proposed scheme can be summarized as follows. Given that the underlying graph structure is known *a priori*, the algorithm can be run before the sensors start to collect any data and they can thus be preconfigured. Although very infrequently, the graph structure may change over time; it is then important to update the sampling scheme accordingly. However, how to do that efficiently is a subject of future work.

The method works well for cases where the graph structure can be induced from some physical aspects of the setup. An example of this is the case of a network of water pipes, here the connections of the pipes gives a natural way of connecting the nodes. Because the sampling scheme is derived before deployment and all sensors sample periodically, as seen from the sensor perspective, the strain on the sensors from computations is practically nonexistent. To reconstruct the entire dataset we must assume that the server has a reasonable amount of computation power and memory available and that we allow for some error ε.

5 Conclusion

We have considered the problem of energy-efficient sampling of IoT sensors that are deployed in a system with an underlying graph structure. We leverage this structure through a graph signal processing framework to obtain the maximal number of disjoint sets to be sampled, while the the expected mean square error for each set is below a given error threshold. We have evaluated the algorithm in a scenario where the

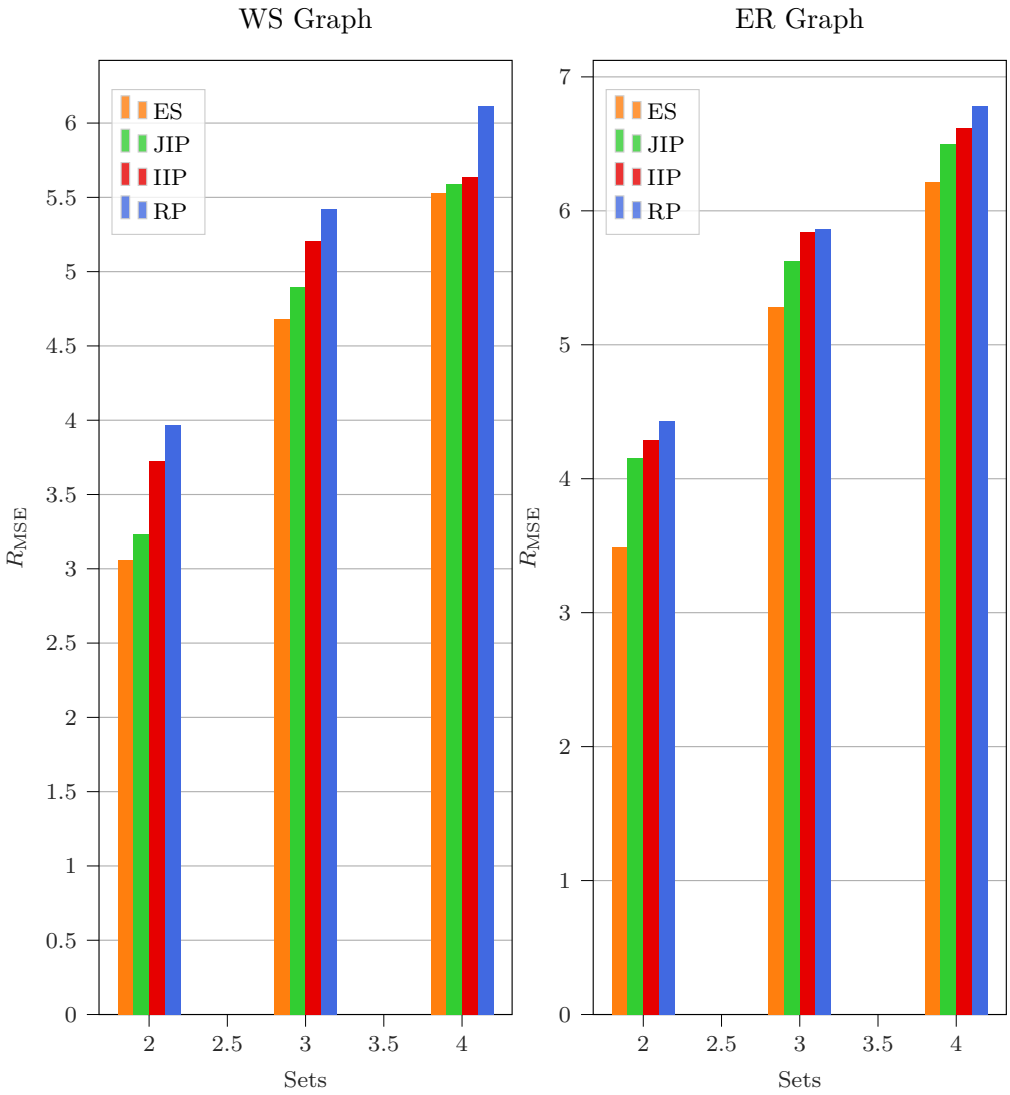


Fig. A.4: Exhaustive search for best partition on two graphs with 15 nodes.

underlying graph is build according to a water network. The results show that JIP gives a lower error than IIP and RP. Although the gains are not particularly high, the methodologically novel approach shows promises through its consistent improvement over the other approaches and incurs no additional computational cost. Furthermore, our results show that the gain from using our scheme increases for more clustered graphs like the WS small-world model, making JIP particularly suited for those scenarios. As a future work, we will generalize the approach to allow for efficient update of the partitions upon changes in the underlying physical graph.

References

[1] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, and A. Dekorsy, "Massive machine-type communications in 5g: Physical and mac-layer solutions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 59–65, 2016.

[2] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging filed of signal processing on graphs," *IEEE Signal Processing Magazine*, 2013.

[4] G. B. Ribeiro and J. B. Lima, "Graph signal processing in a nutshell," *Journal of Communication and Information Systems*, vol. 33, no. 1, 2018.

[5] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3864–3868.

[6] L. F. Chamon and A. Ribeiro, "Greedy sampling of graph signals," *IEEE Transactions on Signal Processing*, vol. 66, no. 1, pp. 34–47, 2017.

[7] F. Wang, Y. Wang, and G. Cheung, "A-optimal sampling and robust reconstruction for graph signals via truncated neumann series," *IEEE Signal Processing Letters*, vol. 25, no. 5, pp. 680–684, 2018.

[8] B. Güler, A. Jayawant, A. S. Avestimehr, and A. Ortega, "Robust graph signal sampling," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7520–7524.

[9] A. Chiumento, N. Marchetti, and I. Macaluso, "Energy efficient wsn: a cross-layer graph signal processing solution to information redundancy," in *2019 16th*

International Symposium on Wireless Communication Systems (ISWCS). IEEE, 2019, pp. 645–650.

[10] Z. Wei, A. Pagani, G. Fu, I. Guymer, W. Chen, J. McCann, and W. Guo, "Optimal sampling of water distribution network dynamics using graph Fourier transform," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1570–1582, 2019.

[11] L. A. Rossman *et al.*, "EPANET 2: users manual," 2000.

Paper B

Finding Representative Sampling Subsets in Sensor Graphs using Time Series Similarities

Roshni Chakraborty, Josefine Holm, Torben Bach Pedersen, and Petar Popovski

The paper has been submitted to the
ACM Transactions on Sensor Networks 2022.

The layout has been revised.

Abstract

With the increasing use of IoT-enabled sensors, it is important to have effective methods to query the sensors. For example, in a dense network of battery-driven temperature sensors, it is often possible to query (sample) only a subset of the sensors at any given time, since the values of the non-sampled sensors can be estimated from the sampled values. If we can divide the set of sensors into disjoint so-called representative sampling subsets that each represents all the other sensors sufficiently well, we can alternate between the sampling subsets and thus, increase the battery life significantly of the sensor network. In this paper, we formulate the problem of finding representative sampling subsets as a graph problem on a so-called sensor graph with the sensors as nodes. Our proposed solution, SubGraphSample, consists of two phases. In Phase-I, we create edges in the similarity graph based on the similarities between the time-series of sensor values, analyzing six different techniques based on proven time-series similarity metrics. In Phase-II, we propose six different sampling techniques to find the maximum number of representative sampling subsets. Finally, we propose AutoSubGraphSample which auto-selects the best technique for Phase-I and Phase-II for a given dataset. Our extensive experimental evaluation shows that AutoSubGraphSample can yield significant battery life improvements within realistic error bounds.

1 Introduction

Recently, Internet of Things (IoT) enabled sensors are being widely used for different applications, such as, military operations, traffic management, home-service, healthcare, and several others [1, 2]. Irrespective of the application, the sensors generate continuous data, in the form of time-series. This massive production of data has led to new challenges in data processing, storage and analysis [3]. In order to handle this massive information overload, there is a need to develop effective methods that can query the sensors efficiently, for example, to preserve battery life [4, 5]. Identifying disjoint *representative sampling subsets* such that only a *representative sampling subset* is queried at a given time is an efficient solution for quering the sensors. This is possible if the time-series generated by the different sensors are similar and each *representative sampling subset* represents the values of all the sensors sufficiently well [6]. Therefore, in this paper, we aim to identify the maximum number of disjoint *representative sampling subsets* given the sensors and their time-series data.

We now discuss this through a motivating example of a sensor network, SN, which comprises of 9 sensors as $\mathcal{S} = S_1, S_2, \cdots, S_9$ as shown in Figure B.1. Each of these sensors, S_i records the temperature of a particular location as a time-series, T^i with 5 time instances $1, 2, \cdots, 5$ resulting in $T^i_1, T^i_2, \cdots, T^i_5$ for a sensor, S_i. We provide the data recorded by $\mathcal{S}$ similar to existing datasets [7] in Table B.1. In order to understand the

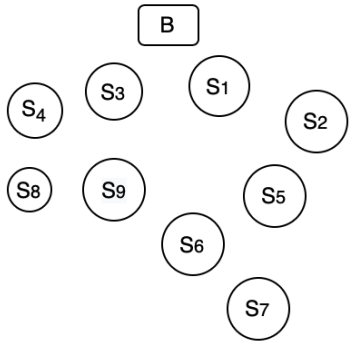


Fig. B.1: An overview of a sensor network, SN, that comprises of sensors, S_1 to S_9, where each of the sensor records data in time-series, T^i. The sensors communicate to the base-station B for processing and computation. The data recorded by sensors, S is shown in Table B.1

similarity between the time-series data among sensors, we compute the *Fast DTW* [8] distance between each pair of sensors (see Table B.2). *Fast DTW* [8] distance is inversely proportional to the similarity between a pair of sensors on the basis of the time-series. Although there are several ways to calculate the similarity or distance between a pair of sensors, we use *Fast DTW* on SN as an example to calculate distance and provide an intuition of how the similarity between sensors can be utilized to identify disjoint *representative sampling subsets*. We discuss *Fast DTW* in detail in Subsection 4.2 and its application on SN in Subsection 4.4. We highlight the results of *Fast DTW* in Table B.2 to give an intuition of the research problem. For example, as shown in Table B.2, S_1 and S_2 are similar with a low distances of 8 whereas S_1 and S_9, S_2 and S_4 are dissimilar with a high distances of 23 and 28, respectively. Therefore, we can query S_1 and S_2 at alternating timestamps, improving battery life of both S_1 and S_2 while still getting sufficiently accurate results. Extending this idea to the entire SN, we identify 3 disjoint *representative sampling subsets*, which each represent all the sensors within a given error bound on the time-series values. The intuition is that by identifying 3 disjoint *representative sampling subsets* from SN, we can increase the battery longevity by 3 times compared to the case of querying all sensors at all times. We represent this in Figure B.2. It is therefore required to devise a system that can perform *a*) creation of a similarity graph of the sensors on the basis of the time-series data and, *b*) identification of the maximum number of *representative sampling subsets* from the similarity graph.

Several selection sampling techniques in graph signal processing domain [9] have been proposed, including randomized [9, 10] or deterministic greedy sampling [11, 12] techniques which focus on finding a single *representative sampling subset*. However, these approaches identify only one sub-set of sensors and therefore, do not solve our objective.

Sensor	T^i	Sensor	T^i
S_1	$[4, 5, 5, 5, 4]$	S_6	$[7, 9, 10, 10, 9]$
S_2	$[6, 6, 7, 7, 5]$	S_7	$[9, 9, 9, 11, 7]$
S_3	$[1, 1, 3, 3, 3]$	S_8	$[0, 3, 3, 3, 0]$
S_4	$[0, 0, 1, 1, 1]$	S_9	$[1, 1, 1, 4, 1]$
S_5	$[8, 8, 8, 8, 6]$		

Table B.1: The time-series of the 9 sensors of SN is shown

S_i	S_j	D_{ij}	S_i	S_j	D_{ij}	S_i	S_j	D_{ij}
S_1	S_2	8	S_1	S_6	16	S_1	S_9	23
S_2	S_5	7	S_2	S_4	28	S_2	S_7	16
S_3	S_1	12	S_3	S_2	20	S_3	S_8	5
S_6	S_5	9	S_6	S_7	6	S_6	S_8	36
S_9	S_8	7	S_9	S_5	30	S_9	S_3	4

Table B.2: The *Fast DTW* distance between few pairs of sensors of SN is shown

Furthermore, existing sampling approaches rely on the availability of the graph topology of the sensors which might not be always available. Therefore, in this paper, we propose a Two-phase framework, namely, *SubGraphSample*, that identifies the maximum number of *representative sampling subsets* on the basis of similarity among the sensors such that sensors that generate similar data belong to different *representative sampling subsets*. In *SubGraphSample*, we initially create a similarity graph of the sensors in Phase-I and then, iteratively identify representatives from each possible subgraph of the similarity graph to form the maximal number of possible *representative sampling subsets* in Phase-II. We highlight our contributions next :

1. We propose *SubGraphSample* that does not require the graph topology of the sensors and can directly identify the maximum number of possible sampling sets given only the readings of the sensors. To the best of our knowledge, there is no existing sampling algorithm that can directly identify the maximum number of sampling sets given only the readings of the sensors. We highlight our major contributions for *SubGraphSample* next.

   (a) To generate the similarity graph of the sensors, we compare and tune 6 different types of similarity approaches. Although these are previously proposed approaches, there is no existing comprehensive research work that compares the performance of these different types of approaches and provide intuition about which approach to use given a dataset. Additionally, most of these approaches are mainly used only to identify the similarity between sensors

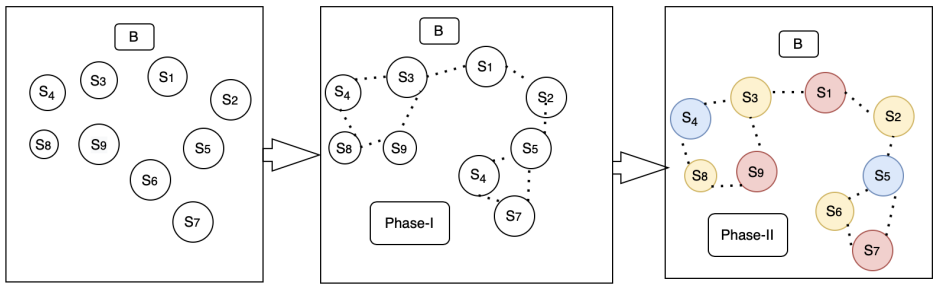


Fig. B.2: An overview of the Two-phase proposed framework, *SubGraphSample* on the sensor network, SN, is shown. In Phase I, we create a similarity graph, $\mathcal{G}$ of the sensors where there is an edge between a pair of sensors if the similarity is greater than the threshold. In Phase II, we identify different *representative sampling subsets* (represented by different colors) from $\mathcal{G}$.

for different applications. We propose an effective utilization of these approaches for similarity graph creation for sampling sets identification which mitigates the dependency of existing sampling approaches on the availability of graph topology. We also provide recommendations of which similarity graph creation approach to use given a dataset.

(b) We propose 6 novel different types of sampling approaches in Phase-II, namely *Strat*, *MSV*, *JIP*, *SIP*, *Frob* and *Par*. In *Strat*, we propose three different approaches which utilizes network science theoretic concepts to identify the maximum number of possible sampling sets. To the best of our knowledge, this is the first research work that proposes network science theoretic concepts for sampling sets identification and is designed specifically for time-series based applications. In *JIP* and *SIP*, we propose utilization of the difference between the similarity graph signal and the signal of a sensor, namely mean square error, MSE, to select the sensor into a sampling set and thereby, iteratively create the maximum number of possible sampling sets. Additionally, in *MSV*, *Frob* and *Par* we extend methods, which focus on identifying one optimum sampling set, to methods generating the maximum number of possible sampling sets. While we explore a measure using the eigen decomposition of the adjacency matrix in *MSV*, we explore different measures using the eigen decomposition of the graph laplacian matrix in *Frob* and *Par*, respectively.

(c) For our experiments, we compare the performance of Phase-I approaches in creating similarity graph given a dataset by the *average path length*, *clustering coefficient*, *edge density*, and measure how well the graph topology represents the dataset by *total cumulative energy residual*. Additionally, we use reconstruction error to compare the performance of Phase-II sampling

approaches and *SubGraphSample* respectively. Our experimental evaluations on 4 datasets show that the best combination of the graph creation approach and sampling approach can provide $5-13$ times increase in battery life with a $20-40\%$ error bound for a given dataset. We discuss our experimental evaluations in detail in Subsection 7.1 and Subsection 7.2 respectively.

2. We propose an auto-tuned approach, *AutoSubGraphSample*, to select the best combination of Phase-I and Phase-II approaches for a given dataset.

3. We evaluate *AutoSubGraphSample* on 9 representative datasets. Our observations indicates that *AutoSubGraphSample* can generalize well to new datasets.

4. We perform several additional experimental analyses, such as, evaluate the performance of *AutoSubGraphSample* when only a fraction of time-series is provided (details in Subsection 7.8) and a comparative analysis of the *AutoSubGraphSample* with the optimum results at each phase (details in Subsection 7.4 and Subsection 7.5).

The organization of the paper is as follows. We discuss the existing research works in Section 2 followed by the problem statement in Section 3. In Section 4 and 5, we discuss the proposed approach and the experiments setup. We show our observations in Section 6. We, finally, draw our conclusions and future works in Section 8.

2 Related Work

2.1 Phase-I: Creation of Similarity Graphs

We categorize the existing research works that could be used to create a similarity graph of sensors, given the data generated by the sensors, into three types of approaches: *Statistical*, *Time-Series Analysis* and *Graph Signal Processing*.

Statistical Approaches

In order to identify graphs between sensors, a simple way is to calculate similarity between each pair of sensors and then, create an edge between them if their similarity is greater than the threshold [13]. Therefore, existing metrics, such as the Pearson correlation, the Jaccard coefficient, the Gaussian radial basis function and mutual information are used to compute the pairwise similarity and thereby, identify the graph topology [14, 15]. Feizi et al. [16] extended the pairwise correlation by including the indirect dependencies from the transitive correlations through a network deconvolution based approach.

Time-Series Analysis based Approaches

Using time-series data, one can compute the similarity between a pair of sensors and use it as a basis to define an edge in the graph. The existing approaches based on time-series can be classified into distance/neighbourhood based methods and feature based methods [17]. Distance based methods focus on identifying different distance metrics to align a pair of time-series [18]. Traditional distance metrics that are inspired by the concept of edit distance [19], include, Lp-norms [20], Euclidean Distance [21], Dynamic Time Warping (DTW) [22], Longest Common Sub-sequence (LCSS) [23], Edit Sequence on Real Sequence (EDR) [24], Swale [25], Spatial Assembling Distance (SpADe) [26], etc. Further, several existing research papers have proposed different variants [27] of these traditional distance metrics for different objectives, such as run-time [28], applicability to specific problem [29], etc. There are several feature-based classification algorithms [30], such as, traditional SVM and decision trees, naive logistic model (NL), the Fisher kernel learning (FKL) [31], Hidden State Conditional Random Field [32] and histogram-based method [33, 34], to determine time-series similarity. However, several studies [35, 36] have shown that feature-based approaches fail to capture the inherent intrinsic attributes particular to time-series data which are better captured by distance-based approaches. Further, several recent research works have proposed integration of both neighbourhood-based metrics [17, 37] and distance based metrics to train machine learning models, such as, SVM, Random Forest and ensemble models [38]. Recently, several research papers have proposed different neural network architectures, such as, autoencoders [39], deep networks [40], meta-learning based pre-training [41], attention modules [40, 42] to capture the complex temporal relationships in time-series.

Graph Signal Processing based Approaches

In order to ensure analysis and processing of the graph signals in both the vertex and the spectral domain of the graph, several recent papers infer an optimum graph topology such that the input data form graph signals with smooth variations on the resulting topology [43, 44]. Dong et al. [45] propose a factor analysis based model which was extended by Kalofolias et al. [46] to include sparsity. However, these approaches assume that the graph signals used for training are smooth.

Summary of Insights

Considering the variety of existing approaches to infer the similarity graph based on the sensing data, there is still a lack of a comprehensive study that compares how different approaches perform on a given dataset. These approaches have mainly been used for different applications, such as, to infer similarity or distance between two time-series, create an optimum graph topology given the sensor signals, etc. In addition, to the best of our knowledge, there is no existing research that utilizes this similarity to generate

a similarity graph for sampling approaches. Therefore, there is no existing research tailored to identify the maximum number of disjoint representative sampling subsets given the time-series data of the sensors. In this paper, we select several prominent existing approaches from the three categories described above and compare them in their role in Phase-I for a given dataset.

2.2 Phase-II: Sampling Algorithms

Randomized sampling based approaches [9, 10, 47] select nodes from a predetermined probability distribution. They have a low computational cost, but cannot ensure the same quality at each selection. Deterministic greedy sampling techniques resolve this by selecting the optimal sensor at each iteration. This deterministic operation scales with polynomial complexity [11, 12]. However, most of these sampling techniques search for only one optimal sampling set and do not consider the time dimension of the data [48–51]. Therefore, these techniques do not resolve our objective of maximizing battery longevity. The works [52, 53] identify each sampling set representing a time-graph signal; nevertheless, the same node may participate in different *representative sampling subsets* which is not suitable to maximize the battery longevity. The sampling technique from [54] can ensure improvement in battery lifetime. However, [55] has shown that the approach from [54] is sub-optimal. Although Gedik et al. [56] and Liu et al. [57] intend to maximize the battery longevity of the sensor network, these techniques are not applicable for the application we are interested in. For example, Gedik et al. [56] assume that the set of clusters is given and therefore, requires both the similarity among sensors and the clustering information to be provided beforehand. Although Liu et al. [57] do not require the clustering information to be provided, it requires the location of the sensors to be provided and clusters the sensor network based on the spatial correlation among sensors. In comparison, we only require the time-series data of the sensors. Thus, we propose six different sampling techniques to identify the maximum number of *representative sampling subsets* of which we propose two novel and extend four existing ones [11, 58, 59].

3 Problem Statement and Framework

3.1 Problem Statement

Given a sensor graph comprising n IOT-enabled sensors, $\mathcal{S} = (S_1, S_2, \ldots, S_n)$, the time-series data for the sensor S_i is denoted by T^i. Let $\mathcal{CP}$ denote the set of all (say, q in this case) possible complete partitions of the network $\mathcal{CP} = (\mathcal{SP}_1, \mathcal{SP}_2, \ldots, \mathcal{SP}_q)$. A partition, $\mathcal{SP}_u$ consists of several non-empty subsets of $\mathcal{S}$, i.e., $\mathcal{SP}_u = (SP_u^1, SP_u^2, \ldots, SP_u^k)$ such that $\bigcup_{i=1}^{k} SP_u^i = \mathcal{S}$. The sensors are battery-powered and have low computing

power. We also assume that the time-series data have no missing values. We identify the optimum partition, OSP, from all the possible complete partitions, $\mathcal{CP}$ such that,

$$\begin{aligned} & OSP = \text{argmax}_{x \in \mathcal{CP}}(|\mathcal{SP}_x|) \\ s.t. \quad & Error(\mathcal{SP}_x^i, \mathcal{S}) \leq \varepsilon \quad \forall \quad \mathcal{SP}_x^i \in \mathcal{SP}_x \\ & \mathcal{SP}_x^i \cap \mathcal{SP}_x^j = \emptyset \quad \forall \ (\mathcal{SP}_x^i \neq \mathcal{SP}_x^j) \in \mathcal{SP}_x \end{aligned} \tag{B.1}$$

The optimum sampling partition, OSP, is the partition that comprises of the maximum number of *representative sampling subsets*, $\mathcal{SP}_u$, such that each of these non-empty subsets, $\mathcal{SP}_u^i$ can represent the values of all sensors well enough, i.e., the error in the information recorded by $\mathcal{SP}_u^i$ when compared to $\mathcal{S}$ must be less than the threshold, ε, as $(Error(\mathcal{SP}_u^i, \mathcal{S}) \leq \varepsilon)$. We consider reconstruction error to calculate $Error(\mathcal{SP}_u^i, \mathcal{S})$ which we discuss in details in Subsection 6.2. Additionally, we assume only periodic round robin scheduling of each *representative sampling subset*, $\mathcal{SP}_u^i$, of OSP in this paper. Furthermore, we consider constraint that no two subsets of OSP can overlap, i.e., $\mathcal{SP}_u^i \cap \mathcal{SP}_u^j = \emptyset$.

This problem can be reformulated to identify the optimal partition OSP that minimizes the error for a given number K *representative sampling subsets*:

$$\begin{aligned} & OSP = argmin_{x \in \mathcal{CP}}(Error(\mathcal{SP}_x, \mathcal{S})) \\ s.t. \quad & |\mathcal{SP}_x| = K \\ & \mathcal{SP}_x^i \cap \mathcal{SP}_x^j = \emptyset \quad \forall \ (\mathcal{SP}_x^i \neq \mathcal{SP}_x^j) \in \mathcal{SP}_x \end{aligned} \tag{B.2}$$

We show a brute-force solution to Equation B.1 in Algorithm 1 where we initially identify all the possible complete partitions of $\mathcal{G}$. We, then, enumerate all of these complete partitions to identify that partition which has the maximum number of sampling partitions such that each sampling partition is within the error threshold. Identifying one complete partition of a graph is NP-hard [60] (Algorithm 1, line num 1).

The number of possible complete partitions in a $\mathcal{G}$ of n sensors are given by the Bell numbers B_n $(B_0 = 1, \dots)$[18] (where, $B_0 = 1$, $B_1 = 1$, $B_2 = 2$, $B_2 = 5$, $B_4 = 15$, $B_5 = 52$, etc.) [61]. Therefore, Therefore, the time-complexity of the inner loop of Algorithm 1 (lines $3 - 18$) is $\mathcal{O}(B_n s^2)$, where s represents the maximum possible size of a sampling set, $\mathcal{SP}_x^i$ and B_n is number of possible complete partitions of $\mathcal{CP}$ [61]. Therefore, Algorithm 1 given $\mathcal{CP}$ (from line 3-16) is exponential with respect to the number of sensors, n [62]. Therefore, we instead propose a heuristic solution based on a 2-phase framework, *SubGraphSample*, which can solve either of the dual problems, Equation (B.1) or Equation (B.2). In Phase-I, we create a similarity graph, $\mathcal{G} = (V, E)$ such that the vertices V are the sensors, S and the edges E represent the similarity of the recorded data between each pair of sensors, S_i and S_j using existing approaches. In Phase-II, we identify the OSP from $\mathcal{G}$. An overview of the proposed approach on SN is shown in FigureB.2. We discuss graph creation approaches for Phase-I in Section 4

Algorithm 1 *Brute-Force Solution*

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$ and error threshold, ε
Output Optimum Sampling Partition OSP

Find the set of all possible complete partitions, $\mathcal{CP} = (\mathcal{SP}_1, \mathcal{SP}_2, \ldots, \mathcal{SP}_q)$ of $\mathcal{G}$
Intialize max=0 ▷ variable to store the maximum number of sampling sets among all partitions
for x in $\mathcal{CP}$ **do** ▷ iterate through all the possible complete partitions
    Intialize count=0 ▷ variable to store the number of sampling sets of $\mathcal{SP}_x$ within ε
    for i in $\mathcal{SP}_x$ **do** ▷ $\mathcal{SP}_x$ is a complete partition of $\mathcal{CP}$ and $\mathcal{SP}_x^i$ is i^{th} sampling set of $\mathcal{SP}_x$
        if $(Error(\mathcal{SP}_x^i, \mathcal{S}) \leq \varepsilon)$ **then**
            count=count+1
        Intialize inter=0 ▷ inter is 1 if $\mathcal{SP}_x^i$ is non disjoint
        for j in $\mathcal{SP}_x$ **do**
            if $(\mathcal{SP}_x^i \cap \mathcal{SP}_x^j)!=\emptyset$ **then**
                inter=1
    if $(count==|\mathcal{SP}_x|)$ and (inter==0) and $(|\mathcal{SP}_x| \geq max)$ **then**
        max=$|\mathcal{SP}_x|$
        $OSP = \mathcal{SP}_x$ ▷ $\mathcal{SP}_x$ has the maximum number of sampling sets among all members of $\mathcal{CP}$ examined
Return OSP

and propose sampling approaches for Phase-II in Section 5. Furthermore, we propose Algorithm *AutoSubGraphSample* to recommend the most suitable algorithm for both Phase-I and Phase-II, respectively, for a given dataset.

3.2 Preliminaries

We now discuss the graph signal processing preliminaries needed to understand the proposed sampling techniques and evaluation metrics. We consider a dataset which comprises of n sensors, t as the length of the time-series of each sensor such that $T = 0, 1, \ldots, (t-1)$, $S = S_0, S_1, \ldots S_{n-1}$ is the set of sensors and s as signal for the rest of the paper. We provide a summary of the notations in Table B.3.

- *Degree Matrix, D:* A diagonal matrix that contains the degree of each node, i.e., with entries $D_{ii} = \sum_{j=1}^{n} \mathcal{A}_{ij}$ and $D_{ij} = 0$ for $i \neq j$, where $\mathcal{A}$ is the adjacency matrix.
- *Graph Laplacian, L* : L is calculated as $L = \mathcal{A} - D$, where $\mathcal{A}$ is the adjacency

Notation	Description
$\mathcal{S} = (S_1, S_2, \ldots, S_n)$	list of sensors
n	number of sensors in $\mathcal{S}$
m	length of time-series of a sensor, S_i
K	number of representative sampling subsets
$\mathcal{SP}_u$	u^{th} sampling partition
OSP	optimum sampling partition
D	degree matrix
L	graph laplacian
Λ	eigenvalues of L
V	eigenvectors of L
x^r	graph signal, x at r^th time-stamp where $r \in 0, 1, \ldots, (t-1)$
$\mathcal{A}$	adjacency matrix
Σ	eigenvalues of $\mathcal{A}$
U	eigenvectors of $\mathcal{A}$
Σ	eigenvalues of $\mathcal{A}$
U	eigenvectors of $\mathcal{A}$
$\hat{x}^k$	Graph Fourier Transform of x^k
$\mathcal{SP}^i_u$	i^{th} sampling set of $\mathcal{SP}_u$
ε	error threshold
$\mathcal{G}$	similarity graph
P_y	similarity graph creation approach, y
$\mathcal{G}_y$	similarity graph created by y
$\rho(S_i, S_j)$	*Pearson Correlation Coefficient*, of S_i and S_j
$NodeScore(S_i)$	importance of S_i
R_i	relevance of S_i
C_k	k^{th} community of $\mathcal{G}$
$IG(S_i, \mathcal{SP}^i_x)$	information gain provided by S_i with respect to $\mathcal{SP}^i_x$
E_d	edge density of $\mathcal{G}$
$SamSErr(\mathcal{SP}^i_x, \mathcal{S})$	sampling error of $\mathcal{SP}^i_u$ wrt to $\mathcal{S}$
$Err(OSP, S)$	reconstruction error of $\mathcal{SP}_u$ calculated for K sampling sets
Avg P_l	*average path length* of $\mathcal{G}$
Avg CC	*clustering coefficient* of $\mathcal{G}$
$TCER$	total cumulative energy residual

Table B.3: A summary of the notations

matrix and D is the degree matrix [63].

- *Signal* : A signal represents a time-dependent function that conveys information [64]. For example, the signal is $s_T = s_0, s_1, \ldots, s_{t-1}$ and s_i is a sample of the signal s_T.
- *Graph Signal, x* : A signal whose samples are indexed by the nodes of a graph [64]. In this paper, we consider graph-time signal, i.e., one graph signal, x^r per time stamp, r where $r \in 0, 1, \ldots, (t-1)$. Therefore, a graph signal represents the values of each sensor at a time-stamp, i.e., x^r comprises of n samples (for n sensors) where each sample x^r_j is the value for sensor, S_j at the time stamp r.
- *Smoothness*: A graph signal x^r at the $r-$th time-stamp is smooth if it has similar values for the neighbouring nodes of $\mathcal{G}$.
- *Graph Fourier transform, GFT* : GFT is the eigendecomposition of the graph Laplacian, L or adjacency matrix, $\mathcal{A}$ into eigenvalues, Λ and eigenvectors, V. The eigendecomposition of L is $L = V\Lambda V^{-1}$. GFT of x^k, i.e., $\hat{x}^k$ which is defined as $\hat{x}^k = V^{-1}x^k$ [63].
- *Bandlimited Signal* : This is a signal that is limited to have non-zero spectral density only for frequencies that are below a given frequency. If $\hat{x}^k$ is bandlimited i.e. there exist a $\mathcal{B} \in \{0, 1, ..., F-1\}$ such that $\hat{x}_i = 0$ for all $i \geq \mathcal{B}$, then x^k is compressible and can be sampled [64].

4 Phase I : Similarity Graph Creation

In this Section, we discuss the creation of the similarity graph, $\mathcal{G}$, by selecting different approaches from Statistical Approaches, Time-series based Approaches and Graph Signal Processing based Approaches.

4.1 Statistical Approaches

We discuss 2 statistical approaches next.

Correlation-based Approach, P_{corr}

From [13], we calculate *Pearson Correlation Coefficient*, $\rho(S_i, S_j)$ to determine the similarity between S_i and S_j as:

$$\rho(S_i, S_j) = \frac{\sigma(S_i, S_j)}{\sqrt{var(S_i) \cdot var(S_j)}} \tag{B.3}$$

where, $\sigma(S_i, S_j)$ is the co-variance between S_i and S_j and $var(S_i)$ calculates the variance of the data for S_i. Therefore, we create an edge between S_i and S_j in $\mathcal{G}_{corr}$ if $\rho(S_i, S_j)$ is greater than the threshold. The time complexity of P_{corr} is $\mathcal{O}(n^3)$ [13].

Network Deconvolution P_{conv}

We use network deconvolution [16, 65] to create $\mathcal{G}_{conv}$ from the adjacency matrix $\mathcal{A}$. Network deconvolution calculates $\mathcal{A}$ based on the co-variance matrix, Σ, determined from the data S generated by the sensors:

$$\mathcal{A} = \Sigma(\mathbb{I} + \Sigma)^{-1} \tag{B.4}$$

The time complexity of P_{conv} is $\mathcal{O}(n^3)$ [16].

4.2 Approaches based on Time-Series

We discuss 3 approaches that determine similarity based on the time-series of each pair of sensors, S_i and S_j.

Dynamic Time Warping (DTW), P_{dtw}

P_{dtw} measures the distance between a pair of sensors, S_i and S_j by calculating the distance, $DisDTW(S_i, S_j)$ based on the Euclidean distance of the respective time-series of the sensors, S_i and S_j at the particular time-stamp and the minimum of the cumulative distances of adjacent elements of the two-time-series. However, P_{dtw} incurs high computational cost which runs across different time-series. Therefore, we use *Fast DTW* [8] which being an approximation of P_{dtw} runs in linear time and space [8]. We calculate the distance between S_i and S_j as $DisFDTW(S_i, S_j)$ and create an edge between S_i and S_j in $\mathcal{G}_{dtw}$ if the $DisFDTW(S_i, S_j)$ is less than the threshold. The time complexity of P_{dtw} is $\mathcal{O}(n^2m)$ as the calculation of $DisFDTW(S_i, S_j)$ is $\mathcal{O}(m)$ [8] (m being the length of the time-series) and we calculate $DisFDTW(S_i, S_j)$ for n pairs of sensors to create $\mathcal{G}_{dtw}$.

Edge Estimation based on Haar Wavelet Transform, P_{haar}

The data generated from the sensors is inherently unreliable and noisy. Therefore, we compress the time-series of sensor, S_i to effectively handle the unreliability in the data by *Haar wavelet transform* [66]. We select the K-largest coefficient for S_i and S_j to get a compressed approximation as S_i' and S_j' respectively [67]. We create $\mathcal{G}_{haar}$ in which an edge between S_i and S_j exists if the *Euclidean distance* between S_i' and S_j' is less than the threshold. The time complexity of P_{haar} is $\mathcal{O}(n^2 log_2 m)$ as the calculation of Haar distance between a pair of sensors is $\mathcal{O}(log_2(m))$ [68] (m being the length of the time-series) and we calculate the distance for n pairs of sensors to create $\mathcal{G}_{haar}$.

K-NN Approach, P_{nei}

We follow *K nearest neighbours*, where a class of a node is assigned on the basis of its K nearest neighbours [69]. We initially calculate the distance between a pair of sensors, S_i and S_j based on *Euclidean distance* and create an edge between S_i and S_j in $\mathcal{G}_{nei}$ if the distance between them is among the least *K-distances*. The time complexity of P_{nei} is $\mathcal{O}(mlog_2n)$ [70].

4.3 Approaches based on Graph Signal Processing, P_{gsp}

We follow [46] to infer the graph topology from signals under the assumption that the signal observations from adjacent nodes in a graph form smooth graph signals. The solution from [46] is scalable and the pairwise distances of the data in matrix

$$Z_{i,j} = \|x_i - x_j\|^2$$

are introduced as in

$$W^* = \min_{W} \quad \|W \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log(W\mathbf{1}) + \frac{\beta}{2}\|W\|_F^2 \tag{B.5}$$

W^* is the optimal weighted adjacency matrix, $\|\cdot\|_{1,1}$ is the elementwise 1-norm, $\mathbf{1}^\top \log(W\mathbf{1})$ ensures overall connectivity of the graph by forcing the degrees to be positive while allowing sparsity, α and β are parameters to control connectivity and sparsity respectively. The time complexity to create P_{gsp} is $\mathcal{O}(n^2)$ [46]. We follow the implementation in [71] to determine the weighted adjacency matrix, W. We create an unweighted adjacency matrix, $\mathcal{A}$ and graph, $\mathcal{G}_{gsp}$ by creating an edge in $\mathcal{A}$ and $\mathcal{G}_{gsp}$ if the edge weight in W is greater than threshold. However, we observe that most of the edge weights are around 0 and very few edge weights are within $0.5 - 1$, therefore, it is difficult to create graphs with every edge density by P_{gsp}.

4.4 Summary of Insights

Therefore, in Phase-I, we explore a variety of approaches to generate the similarity graph. These approaches vary immensely on the basis of their methodology. Additionally, to the best of our knowledge, there is no existing research work that studies these different approaches and provide insights on the applicability of these approaches to generate the similarity graph topology for a sampling technique. We experimentally analyze the performance of these approaches in Section 7 to provide heuristics to select the best approach given a dataset. On the basis of the time complexity of these approaches, we observe that P_{nei} performs the best. In order to illustrate how the similarity graph is created given the time-series, we show how P_{dtw} works on SN. On the basis of distance calculated between each pair of sensors as shown in Table B.2, we

create an edge between each pair of sensors, S_i and S_j in $\mathcal{G}_{dtw}$ if the distance between S_i and S_j is less than the threshold, say 15 for SN. Therefore, we show $\mathcal{G}_{dtw}$ in Phase-I of the Figure B.2 where S_1 and S_2, S_3 and S_9 are connected as the distances are 8 and 4 which are less than 15. Additionally, S_1 and S_9, S_2 and S_4 with distance 23 and 28 are not connected.

5 Phase II: Identifying Optimum Sampling Partition (OSP)

We propose several sampling approaches that utilize $\mathcal{G}$ to identify *representative sampling subsets* are representative of the values of all sensors.

5.1 Network Stratification based Approach, *Strat*

We propose a network stratification based sampling approach, *Strat*, that captures the inter-relationship among sensors at group level to inherently handle the sparsity at individual connections and the generic global attributes at the network level[1]. Therefore, in *Strat*, we initially group similar sensors together into communities by Modularity Maximization [72] followed by selecting representatives from each of these communities to create a *representative sampling subset*. We use Modularity Maximization based community detection to group similar sensors as it is similar to the problem of community detection in large networks, as in online social networks [73, 74]. Among the multiple available community detection algorithms, we have opted for Modularity Maximization as it is efficient and scalable to large networks [72]. In order to create a *representative sampling subset*, we select a sensor from each community based on their importance to that community. We denote the importance of a sensor, S_i by *NodeScore(S_i)*. We propose three different mechanisms to calculate *NodeScore(S_i)* which we discuss next in detail. Therefore, we iteratively select sensors from each community in the decreasing order of *NodeScore(S_i)* to form a *representative sampling subset*. We tune the selection method depending on whether we solve Equation (B.1) or Equation (B.2).

Selection by Relevance, *SRel*

In *SRel*, we calculate *NodeScore(S_i)* as the relevance of S_i, i.e., R_i, with respect to C_k to capture the centrality of S_i. Centrality score of a node measures the importance of that node in the network [75]. In this scenario, R_i measures the importance of S_i with respect to all the other sensors present in C_k. Since, an edge between any two sensors in C_k represents the similarity between the time-series of those two sensors, R_i captures the similarity of S_i with all the other sensors of C_k. Therefore, by selecting the sensor with

[1] https://en.wikipedia.org/wiki/Level_of_analysis

Algorithm 2 *SRel*

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K

Output Optimum Sampling Partition OSP

C = Modularity Maximization algorithm($\mathcal{G}$)
$ComMem = \min_{\forall v \in C} Size(v)/K$
for u in S **do**
    $NodeScore(S_u)$ = Eigenvector Centrality(S_u)
Initialize OSP=[]
for i in range(0,K) **do**
    for y in C **do**
        for z in $ComMem$ **do**
            $MNodeScore = \max_{\forall u \in S} NodeScore(S_u)$
            Identify S_u with $MNodeScore$
            Add S_u to $\mathcal{SP}_x^i$
    Add $\mathcal{SP}_x^i$ to OSP
Return OSP

the highest centrality we can ensure maximum representation of all the sensors present in C_k. We utilize the concept of centrality to generate the maximum number of possible *representative sampling subsets*. We measure R_i by Eigenvector Centrality [76]. We determine the number of sensors to be selected from C_k by *ComMem*. The calculation of *ComMem* varies based on whether we solve Equation (B.1) or (B.2). For Equation B.1, we select the minimum number of possible nodes from each community such that the selected nodes can represent all the nodes from the community sufficiently well. We set *ComMem* to be 1 and then, iteratively select *ComMem* sensors from C_k in decreasing order of *NodeScore(S_i)* to create a *representative sampling subset* such that the error of the *representative sampling subset* with respect to all the sensors is less than ε. We repeat these steps to create the maximum number of possible *representative sampling subsets*, i.e., OSP and thus, optimize Equation (B.1).

To solve Equation (B.2), we set *ComMem* as the ratio of the size of the smallest community and the given number of *representative sampling subset*s, K. We, then, iteratively select *ComMem* sensors from a C_k in decreasing order of *NodeScore(S_i)* to form a $\mathcal{SP}_x^i$ and repeat this step for K times to create OSP. We show the algorithm of *SRel* in Algorithm 2. The time complexity of *SRel* depends on the time complexity of the community detection algorithm ($\mathcal{O}(E)$) [72, 77], calculation of eigen-vector centrality ($\mathcal{O}(n^2)$) [76] and calculation and finding the sensor with the highest *NodeScore(S_i)* ($\mathcal{O}(|C|log_2(|C_k|))$). Therefore, the calculation of eigen-vector centrality is most com-

putationally expensive for *SRel* which makes the time complexity of *SRel* be $\mathcal{O}(n^2)$. We follow the same procedure as *SRel* in *SMMR* and *SEMMR* to calculate *ComMem* and determine *OSP* for either (B.1) or (B.2). However, we calculate *NodeScore(S_i)* differently in *SMMR* and *SEMMR* which we discuss next.

Selection by Maximum Marginal Relevance, *SMMR*

In *SMMR*, we consider both relevance and information gain of a sensor to calculate *NodeScore(S_i)*. We propose Maximum Marginal Relevance [78] based score to calculate *NodeScore(S_i)* which is the weighted average of the relevance, R_i and the information gain provided by S_i with respect to $\mathcal{SP}_x^i$, $IG(S_i, \mathcal{SP}_x^i)$. We measure R_i as in *SRel* and $IG(S_i, \mathcal{SP}_x^i)$ as the difference between the adjacency list of S_i and the adjacency list of the already selected sensors in $\mathcal{SP}_x^i$. Therefore, we select the sensor with maximum node score, *MNodeScore* and further, repeat this for *ComMem* times for each C_k iteratively.

Algorithm 3 *SMMR*

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K

Output Optimum Sampling Partition OSP

$ComMem = \min_{\forall v \in C} Size(v)/K$
for u in S **do**
    NodeScore(S_u) = Eigenvector Centrality(S_u)
Initialize sampling subsets, OSP=[]
for i in range(0,K) **do**
    for y in C **do**
        for z in *ComMem* **do**
            $A[S_u]$ = *Adjacency List* of Node S_u
            $D(a, b)$ = *Difference* between a set a and a set b
            Calculate $I_G(S_u, \mathcal{SP}_x^i) = |D(A[S_i], A[\mathcal{SP}_x^i])|$
            Calculate $MNodeScore$ by Equation B.7
            Identify sensor, S_u with $MNodeScore$
            Add S_u to $\mathcal{SP}_x^i$
    Add $\mathcal{SP}_x^i$ to OSP
Return OSP

The calculation of *MNodeScore* is as follows

$$MNodeScore = \max_{\{S_i \in C_k\}} [NodeScore(S_i)] \tag{B.6}$$

$$= \max_{S_i \in C_k} [\beta R_i - (1-\beta) I_G(S_i, \mathcal{SP}^i_x] \tag{B.7}$$

where β is the weight for relevance and $(1-\beta)$ for information gain respectively. For our experiments, we consider β as 0.5 to give equal importance to both relevance and information gain respectively. We show the pseudocode of *SMMR* in Algorithm 3. The time complexity of *SMMR* is similar to *SRel* except for the calculation of Maximum Marginal Relevance [78] which has a time complexity of $\mathcal{O}((|C_k|)^2)$. However, as the number of sensors in a community, i.e., $(|C_k|)$ is much less than the number of sensors in the network, i.e., n, the time complexity for eigen-vector centrality is the most computationally expensive for *SMMR*. Therefore, the time complexity for *SMMR* is same as for *SRel*, i.e., $\mathcal{O}(n^2)$.

Selection by Error based Maximum Marginal Relevance, *SEMMR*

In *SRel* and *SMMR*, we consider the edges between the sensors in $\mathcal{G}$ to calculate *NodeScore(S_i)* and do not consider the actual data generated by S_i. In *SEMMR*, we incorporate this information by calculating $IG(S_i, \mathcal{SP}^i_x)$ as the average of the minimum square error between the data generated by S_i and the other sensors already selected in $\mathcal{SP}^i_x$. We follow the same procedure of *SMMR* to calculate *MNodeScore* and finally, follow the same procedure as discussed in *SRel* to determine *ComMem* and resolve either Equation (B.1) or Equation (B.2) accordingly. Algorithm 4 shows the pseudocode of *SEMMR*. The time complexity of *SEMMR* is same as for *SMMR*, i.e., $\mathcal{O}(n^2)$ as it follows same approach as *SMMR* except for utilizing the time-series of sensors instead of the adjacency list. *SEMMR* is redundant when compared to all the proposed sampling techniques as it also requires time-series of the sensors and is not just dependent on the $\mathcal{G}$.

5.2 Minimum singular value based approach, *MSV*

Chen et al. [58] proposed a greedy-selection based sampling algorithm in which they iteratively select the node that maximizes the minimum singular value of the eigen vector matrix under the assumption that the signal is bandlimited. Under the assumption of bandlimitedness, selection of the best $|\mathcal{B}|$ nodes ensures almost complete reconstruction of the graph signal given that there is no sampling noise. Therefore, by choosing the node that maximizes the minimum singular value, Chen et al. optimize the information in the graph Fourier domain and form a greedy approximation of the best $|\mathcal{B}|$ nodes. The authors consider eigen decomposition of the adjacency matrix, $\mathcal{A} = U\Sigma U^{-1}$, for

Algorithm 4 *SEMMR*

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K

Output Optimum Sampling Partition OSP

$ComMem = \min_{\forall v \in C} Size(v)/K$
for u in S **do**
    $NodeScore(S_u)$ = Eigenvector Centrality(S_u)
Initialize sampling subsets, OSP=[]
for i in range(0,K) **do**
    for y in C **do**
        for z in $ComMem$ **do**
            $T[S_u]$ = *Time-Series Data* of S_u
            $Er(S_u, \mathcal{SP}^i_x)$ = *Average Minimum Square Error* of S_u wrt $\mathcal{SP}^i_x$
            Calculate $I_G(S_u, \mathcal{SP}^i_x) = Er(T[S_u], T[\mathcal{SP}^i_x])$
            Calculate *MNodeScore* by Equation B.7
            Identify sensor, S_u with *MNodeScore*
            Add S_u to $\mathcal{SP}^i_x$
    Add $\mathcal{SP}^i_x$ to OSP
Return OSP

graph Fourier transform, $\hat{x} = U^{-1}x$ and create only one *representative sampling subset* with $|\mathcal{B}|$ nodes by selecting the nodes iteratively according to:

$$m = \text{argmax}_q \sigma_{min}(U_{\mathcal{B},SS_p+\{q\}}) \tag{B.8}$$

where $U_{\mathcal{B},SS_p}$ is the first $|\mathcal{B}|$ rows of U, SS_p represents the set of columns of U and $\sigma_{min}(U)$ is the function for the minimal singular value of U. In this paper, we propose *MSV* which is an extension of [58] where we generate K *representative sampling subsets* by iteratively adding nodes to each *representative sampling subset* according to Equation (B.8) until all nodes have been assigned. We provide the pseudocode of *MSV* in Algorithm 5. The time complexity of *MSV* is ($\mathcal{O}(n^3K^2)$ as it depends on the time complexity of singular value decomposition of the matrix, $U_{\mathcal{B},OSP_p+\{q\}}$ which is $\mathcal{O}(nK^2)$) [79] and we calculate the singular value decomposition for n^2 sensors. Applying *MSV* for *SN* generates 3 *representative sampling subsets* results in (5,4,7), (2,6,3) and (0,8,1). In this paper, we consider $B = V$.

Algorithm 5 Minimum Singular Value, *MSV*

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K, bandwidth, $\mathcal{B}$

Output Optimum Sampling Partition, *OSP*

1: Initialize OSP as a set of K empty sets
2: Initialize $\mathcal{A}$ as the adjacency matrix of $\mathcal{G}$
3: Calculate eigendecomposition $\mathcal{A} = U\Sigma U^{-1}$
4: **for** i in the number of nodes **do**
5: $p = i$ **mod** K
6: $m = \text{argmax}_q \sigma_{min}(U_{\mathcal{B},OSP_p+\{q\}})$
7: $OSP_p \leftarrow OSP_p + \{m\}$
8: Return OSP

5.3 Greedy MSE Based Approach, *JIP and SIP*

We propose two sampling techniques, i.e., Joint Iterative Partitioning, *JIP*, and Simultaneous Iterative Partitioning, *SIP*, that consider *Mean Square Error* to select a node into a *representative sampling subset*. By considering *Mean Square Error*, we ensure that each *representative sampling subset* generated can reconstruct the original graph within an error bound. We discuss *JIP* and *SIP* in detail next. We estimate *MSE* as in [11]:

$$\text{MSE}(OSP_p) = Tr[Q(OSP_p)] \tag{B.9}$$

where

$$Q(OSP_p) = V_{\mathcal{B}} \left(\Lambda^{-1} + \sum_{i \in OSP_p} \eta_i^{-1} v_i v_i^H \right)^{-1} V_{\mathcal{B}}^H \tag{B.10}$$

Here OSP_p is the p'th *representative sampling subset*, $V_{\mathcal{B}}$ is the first $|\mathcal{B}|$ columns of the eigenmatrix, V, v_i is the i'th row of V and η_i is the i'th entry in η which is the variance of the noise. Therefore, MSE is calculated iteratively as nodes are added to a *representative sampling subset*. The reformulation is thoroughly described in [55] as:

$$\text{MSE}(OSP_{p,j} \cup v_s) = Tr[Q_j] - \frac{v_s^H Q_j V_{\mathcal{B}}^H V_{\mathcal{B}} Q_j v_s}{\eta_s + v_s^H Q_j v_s}, \tag{B.11}$$

where

$$Q_j = Q_{j-1} - V_{\mathcal{B}}^H V_{\mathcal{B}} \frac{Q_{j-1} v_u v_u^H Q_{j-1}}{\eta_u + v_u^H Q_{j-1} v_u}, \tag{B.12}$$

$Q_0 = \Lambda$ and u is the index for the most recently added node.

$$MSE(v_s) = MSE(\emptyset \cup v_s) \tag{B.13}$$

In *JIP*, we iteratively create *representative sampling subsets* such that the MSE of each *representative sampling subsets* is within the MSE threshold. Therefore, we initially create a *representative sampling subset* by adding the node with the least MSE according to Equation (B.13) until the MSE of that *representative sampling subset* is less than the threshold. We, further, repeat this for the maximum number of possible *representative sampling subsets*. If there are any nodes left that can not form a *representative sampling subset* on their own, they are divided among existing *representative sampling subsets*. The pseudocode of *JIP* is shown in Algorithm 6 and the time complexity is $\mathcal{O}(n^2|\mathcal{B}|^2)$ [55]. In the results presented in this paper we have used a uniform random vector of $[0, 0.001]$ for η.

Algorithm 6 Joint Iterative Partitioning, JIP

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K, upper bound of *MSE* ε
Output Optimum Sampling Partition, OSP

Initialize $\mathcal{A}$ as the adjacency matrix of $\mathcal{G}$
Initialize L as the graph Laplacian, $L = \mathcal{A} - D$
Calculate eigendecomposition $L = V\Lambda V^{-1}$
$Q_0 = \Lambda, p = 0, j = 1$
$L_s = MSE(v_s)$
$L\_index = argsort(L)$ (largest first)
while $L\_index \neq \emptyset$ **do**
    $u = pop(L\_index_{-1})$
    $OSP_{p,j} = u$
    while $MSE(OSP_{p,j}) > \varepsilon$ and $L\_index \neq \emptyset$ **do**
        Calculate Q_j according to (B.12)
        for i in L_index **do**
            if $MSE(OSP_{p,j} \cup L_i) < \varepsilon$ **then**
                $OSP_{p,j+1} = \{OSP_{p,j} \cup L_i\}$
                $j = 0, p = p + 1, delete(L\_index = i)$
                goto 7
        $OSP_{p,j+1} = \{OSP_{p,j} \cup \arg\min(MSE(L))\}$
        remove chosen node from L_index, $j = j + 1$
if $MSE(OSP_{-1,-1}) > \varepsilon$ **then**
    Split the nodes in OSP_{-1} among the other sets
Return OSP

In *SIP*, we aim to generate *representative sampling subsets* such that every *representative sampling subset* has similar reconstruction error to Equation (B.2). Given the

number of *representative sampling subsets*, K, at each iteration, *SIP* creates the *representative sampling subsets* simultaneously unlike *JIP*. After sorting the nodes according to Equation (B.13), the K nodes with the lowest MSE are added to the *representative sampling subsets*, such that each *representative sampling subset* has been assigned one node. At each iteration, we add the best node according to Equation (B.11) to the *representative sampling subset* with the largest MSE and repeat this for all the *representative sampling subsets* in the same order. We iterate this until all the nodes are allocated to a *representative sampling subset*. The pseudocode of *SIP* is given in Algorithm 7 and the time complexity is same as *JIP*, i.e., $\mathcal{O}(n^2|\mathcal{B}|^2)$ [55]. The *representative sampling subsets* by *JIP* are (5,4,7), (2,6,3) and (0,8,1) and by *SIP* are (5,1,3), (4,8,0) and (2,6,7) respectively.

Algorithm 7 Simultaneous Iterative Partitioning, SIP

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K

Output Optimum Sampling Partition, OSP

Calculate the eigenvalues, Λ and eigenvectors, V of $\mathcal{G}$
Initialize OSP as a list of K empty sets
$Q = array(\Lambda, K), err = zeros(K)$
$L_s = MSE(v_s)$
$L\_index = [0, ..., N]$
for i in range(K) **do**
    $m = \arg\min(L_s)$
    append m to OSP_i
    $err[i] = L_s[m]$
    delete $L_s[m]$ and $L_{index}[m]$
    Update Q_i according to (B.12)
while $L\_index \neq \emptyset$ **do**
    $j = \arg\max(err)$
    $m = ones(N)\max(L_s)$
    for i in L_index **do**
        $m[i] = MSE(OSP_j \cup v_i)$
    append $\arg\min(m)$ to OSP_j
    $err[j] = \min(m)$
    Update Q_i according to (B.12)
    delete $l_{index} = m$
Return OSP

5.4 Minimum Frobenius Norm, *Frob*, and Maximum Parallelepiped Volume, *Par*

Tsitsvero et al. [59] proposed two different greedy based sampling algorithms, *GFrob* and *GPar*, which are based on eigendecomposition of graph Laplacian, $L = V\Lambda V^{-1}$, under the assumption that the signal is $\mathcal{B}$-bandlimited. *Frob*$'$ aims to find the *representative sampling subset* of size $|\mathcal{B}|$ that minimizes the frobenius norm for the pseudo-inverse for the eigenvector matrix restricted to the first $|\mathcal{B}|$ columns and the rows corresponding to the chosen *representative sampling subset*, i.e.

$$OSP_i = \underset{SS_i \in S : |SS_i| = |\mathcal{B}|}{\arg\min} \|(V_{\mathcal{B},SS_i})^+\|_F, \tag{B.14}$$

where $V_{\mathcal{B},SS_i}$ is and the columns of the arbitrary sampling set A_i of V, V^+ denotes the pseudo-inverse and S is the set of all nodes. However, *GFrob* generates only one *representative sampling subset* by adding $|\mathcal{B}|$ nodes in a greedy manner according to:

$$m = \arg\min_q \sum_j \frac{1}{\sigma_j^2(V_{\mathcal{B},SS_p+\{q\}})}. \tag{B.15}$$

where $\sigma_j^2(V)$ denotes the j'th singular value of V.

Algorithm 8 Minimum Frobenius Norm, *Frob*

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K, bandwidth $\mathcal{B}$
Output Optimum Sampling Partition, OSP
Output Find OSP that minimizes the error given K

1: Initialize OSP as an assembly of K empty sets
2: Initialize $\mathcal{A}$ as the adjacency matrix of $\mathcal{G}$
3: Calculate eigen-decomposition $L = V\Lambda V^{-1}$
4: **for** i in the number of nodes **do**
5: $\quad p = i$ **mod** k
6: $\quad m = \text{argmin}_q \sum_j \frac{1}{\sigma_j^2(U_{\mathcal{B},OSP_p+\{q\}})}$
7: $\quad OSP_p \leftarrow OSP_p + \{m\}$
8: Return OSP

In this paper, we extend *GFrob* as *Frob* to generate K *representative sampling subsets* by adding a node to a *representative sampling subset* on the basis of Equation (B.15) in a round robin manner until all the nodes have been assigned to a *representative sampling subset*. An overview of *Frob* is shown in Algorithm 8 and the time complexity

is ($\mathcal{O}(n^3K^2)$ as it depends on the time complexity of singular value decomposition which is $\mathcal{O}(nK^2)$) [79] and we calculate the singular value decomposition for n^2 nodes. In the results presented in this paper we have used $B = V$. Similarly, *Par*, selects $|\mathcal{B}|$ nodes in a greedy manner according to:

$$m = \arg\min_{q} \prod_{j} \lambda_j(V_{\mathcal{B},SS_p+\{q\}} V^H_{\mathcal{B},SS_p+\{q\}}). \tag{B.16}$$

where $\lambda_j(V)$ is the j'th eigenvalue of V. For *Par*, we follow the same approach as proposed in *Frob* to identify the maximum number of possible *representative sampling subsets* that optimizes Equation (B.16) instead of Equation (B.15). An overview of *Par* is shown in Algorithm 9 and the time complexity is ($\mathcal{O}(n^3K^2)$ as it depends on the time complexity of eigen-value decomposition which is $\mathcal{O}((n/K)^3)$ [80] and the eigen-value decomposition is calculated for n^2 nodes. In the results presented in this paper we have used $B = V$. The *representative sampling subsets* generated by *Frob* and *Par* on $\mathcal{G}_{dtw}$ for SN are same which are (8,2,7), (4,5,3) and (0,6,1) respectively.

Algorithm 9 Maximum Parallelepiped Volume, *Par*

Input Time-Series Similarity Graph $\mathcal{G} = (V, E)$, number of *representative sampling subsets* K, bandwidth $\mathcal{B}$

Output Optimum Sampling Partition, OSP

1: Initialize OSP as an assembly of K empty sets
2: Initialize $\mathcal{A}$ as the adjacency matrix of $\mathcal{G}$
3: Calculate eigen-decomposition $L = V\Lambda V^{-1}$
4: **for** i in the number of nodes **do**
5: $\quad p = i$ **mod** k
6: $\quad m = \text{argmin}_q \prod_j \lambda_j(V_{\mathcal{B},OSP_p+\{q\}} V^H_{\mathcal{B},OSP_p+\{q\}})$
7: $\quad OSP_p \leftarrow OSP_p + \{m\}$
8: Return OSP

5.5 AutoSubGraphSample

We have discussed 6 existing graph creation approaches for Phase-I and proposed 6 sampling approaches for Phase-II. However, the performance of these approaches differ across different datasets as they have different properties. Therefore, there is a need to automatically select the most suitable approach for Phase-I and Phase-II respectively given a dataset. In this Subsection, we propose an Algorithm *AutoSubGraphSample* that considers the meta data of the dataset, such as, number of sensors, n and edge density,

E_d to do this. *AutoSubGraphSample* recommends P_{haar} in Phase-I for any edge density in smaller networks (when n is less than 90) and high edge density (when E_d is greater than 0.40) in large networks (when n is greater than 90). It recommends P_{nei} in Phase-I for large networks (when n is greater than 90) with low edge density (less than 0.40). It recommends *SMMR* or *Frob* when edge density is low and *SRel* or *Frob*, otherwise. Our decision of the threshold for n as 90 and E_d as 0.40 is based on our observations from our experiments which we discuss in Section 6. The pseudocode of *AutoSubGraphSample* is shown in Algorithm 10. We validate the generalizability of *AutoSubGraphSample* to a dataset in Section 7.3 by comparing the performance of *AutoSubGraphSample* and the best manually selected algorithms for Phase-I and Phase-II on 9 representative datasets.

Algorithm 10 AutoSubGraphSample : Recommendation for Phase-I and Phase-II

Input $\mathcal{S} = S_1, S_2, \ldots, S_n$ and time-series data of each sensor, T^i is the time-series of i^{th} sensor

Let, e_d be the desired edge density
if $n < 90$ **then**
    Use P_{haar} in Phase-I
    if $e_d < 0.40$ **then**
        Use *SMMR* or *Frob* in Phase-II
    else
        Use *SRel* or *Frob* in Phase-II
else
    if $e_d < 0.40$ **then**
        Use P_{nei} in Phase-I
        Use *SMMR* or *Frob* in Phase-II
    else
        Use P_{haar} in Phase-I
        Use *SRel* or *Frob* in Phase-II

6 Experimental Setup

In this Section, we describe the datasets used in experiments and discuss the different evaluation metrics.

6.1 Dataset Details and Preprocessing

The datasets used for our experiments are:

- D_{epa}: This dataset comprises of 92 sensors and their edge relationships which is simulated with EPANET [81]. EPANET is a tool for simulating water distribution network.
- D_{temp}: This dataset is based on a sensor network that comprises of 74 sensors and their hourly *temperature* [7].
- D_{pol}: This dataset is based on a sensor network deployed at *Aarhus, Denmark* that comprises of 37 sensors and their *Ozone level* recording [2].
- D_{ws}: We create a *synthetic dataset* of 100 nodes that follows the Watts-Strogatz Model [82] with $\beta = 0.5$. We create the data for each of the sensors such that it is strictly bandlimited in the graph Fourier domain.

6.2 Evaluation Metrics

In this Subsection, we discuss the different metrics that we used to compare the different approaches for Phase-I, Phase-II and their combinations. For Phase-I, we compare the approaches in creating different graph topology given a dataset through *average path length*, *clustering coefficient*, *edge density* and measure how well the graph topology represents the dataset by *total cumulative energy residual*. Furthermore, we use *reconstruction error* to measure the performance of Phase-II and the combination of both. We do not discuss *average path length*, *clustering co-efficient* and *edge density* further as they are well known. We detail how we calculate *reconstruction error* and *total cumulative energy residual* next.

Reconstruction Error

Reconstruction of signals on graphs is a well-known problem [83, 84] that provides an estimation of the whole graph, $\mathcal{G}$ by a *representative sampling subset*. For our experiments, we compare the sampling techniques on the basis of the reconstruction error. Given the OSP, which comprises of K *representative sampling subsets* and t as the length of the time-series, we calculate the reconstruction error of a *representative sampling subset* of OSP, $\mathcal{SP}^i_u$, with the $\mathcal{G}$ by measuring the difference between the signal generated by $\mathcal{SP}^i_u$, $\hat{x}^i$, with respect to the signal of $\mathcal{G}$, x, at a time-stamp, say r as

$$\|x^r - \hat{x}^{ir}\|_2. \tag{B.17}$$

We repeat this for all K and t respectively for each sampling technique. We calculate the reconstruction error of OSP, $Err(OSP, S)$ as the average of the total reconstruction error ($TErr(OSP, S)$) over K *representative sampling subsets*. $TErr(OSP, S)$ is the sum of the average reconstruction error of each *representative sampling subset*, i.e.,

[2] http://iot.ee.surrey.ac.uk:8080/datasets/pollution/index.html

$\mathcal{SP}_u^i$ such that i ranges between 1 to K. We calculate the reconstruction error of a *representative sampling subset*, $\mathcal{SP}_u^i$ as $SamSErr(\mathcal{SP}_u^i, S)$ over t time stamps. Therefore, we calculate $Err(OSP, S)$ as follows :

$$\begin{aligned} SamSErr(\mathcal{SP}_u^i, S) &= \sum_{m=1}^{t} \frac{(\|x^m - \hat{x}_p^{im}\|_2)}{\|x^m\|} \\ TErr(OSP, S) &= \sum_{i=1}^{K} (SamSErr(\mathcal{SP}_u^i, S)/t) \\ Err(OSP, S) &= \frac{TErr(OSP, S)}{K} \end{aligned} \tag{B.18}$$

For our results, we show the quartile of $TErr(OSP, S)$ which represents the reconstruction error by a sampling technique.

Total cumulative energy residual, $TCER$

$TCER$ [85] measures the expected energy given a data set to understand how the graph structure represents the data by total cumulative energy of the data. Total cumulative energy of the data is measured by :

$$\mathcal{T}(X, Q) = \sum_{r=1}^{N} (N + 1 - r)\|q_r^\top X\|^2 \tag{B.19}$$

where Q is an orthogonal basis, $TCER$ can then be calculated as

$$1 - \frac{\mathcal{T}(X, V)}{\sum_{R=1}^{N} \sum_{r=1}^{R} \theta_{r,r}^2} \tag{B.20}$$

where V is the eigen vectors of the graph Laplacian and θ are the singular values of X. The values of $TCER$ are in the range of $[0, 1]$ where a high value indicates that the graph represents the dataset well. We follow [86] for the implementation.

7 Results and Discussions

In this Section, we initially evaluate the performance of the approaches for Phase-I and Phase-II separately followed by the validation of Algorithm *AutoSubGraphSample* on 4 representative datasets. We also analyze which combination of algorithms for Phase-I and Phase-II provides most optimal solutions. Lastly, we evaluate the performance of *SubGraphSample* when the whole time-series is not available and discuss additional experiments, such as studying the *impact of* E_d *on reconstruction error*, *frequency analysis* of $\mathcal{G}$.

Phase-I	E_d	Avg P_l	Avg CC	TCER	Th
$\mathcal{G}_{dtw}$	0.19	inf	0.80	0.95	20
	0.41	inf	0.83	0.96	55
	0.60	inf	0.85	0.98	80
	0.76	1.31	0.91	0.97	120
$\mathcal{G}_{nei}$	0.20	1.79	0.81	0.77	8
	0.40	1.59	0.84	0.79	17
	0.60	1.38	0.84	0.78	28
	0.75	1.247	0.87	0.90	37
$\mathcal{G}_{haar}$	0.18	inf	0.65	0.98	16
	0.40	inf	0.75	0.97	18
	0.60	1.41	0.82	0.96	70
	0.75	0.87	1.25	0.88	160
$\mathcal{G}_{gsp}$	0.39	1.61	0.87	0.92	$-1.111 \cdot 10^{-6}$
	0.22	1.78	0.90	0.93	$-1 \cdot 10^{-7}$
	0.59	1.41	0.89	0.90	$-1.1559 \cdot 10^{-6}$
	0.77	1.23	0.96	0.92	$-1.1576 \cdot 10^{-6}$

Table B.4: E_d, Avg P_l, Avg CC for different values of threshold for Phase-I algorithms for D_{temp} is shown

7.1 Phase-I Results: Comparison of the Similarity Graph Creation Approaches

We evaluate the Phase-I algorithms by analyzing two specific properties of the similarity graph topology, $TCER$ and *reconstruction error*.

Evaluation of the Similarity Graph Topology

In order to analyze the properties of the graphs created by different graph creation approaches, we vary the values of the threshold for each of approaches of Phase-I to create graphs with a specific E_d and then, study the *average path length*, and *clustering coefficient* of these graphs. For our experiments, we consider 4 different edge densities; 0.20, 0.40, 0.60 and 0.75 for all the datasets. We show our observations for D_{temp} in Table B.4. Our observations show that there is a significant variance in the properties of the graphs created by the different approaches even for the same E_d and same *dataset*.

$\mathcal{G}_{dtw}$ is disconnected when the E_d is less than 0.70 and the number of sensors is greater than 70 and when the number of sensors is greater than 90 for any E_d. $\mathcal{G}_{haar}$ is disconnected when the E_d is less than 0.40 and the number of sensors is above 90 and $\mathcal{G}_{nei}$ is always connected irrespective of the number of sensors and E_d. Additionally, analyzing the possible values of threshold for different edge densities, we observe P_{dtw}, P_{haar} and P_{nei} can create graphs with any E_d. However, a very small difference in the values of threshold for P_{gsp}, P_{corr} and P_{deconv} can create graphs with highly different E_d. As previously discussed in Section 4.3, we observe that it is difficult for P_{gsp} to generate graphs of different edge densities given a dataset.

The reason for the performance of P_{corr} and P_{deconv} is that they utilize correlation of the time-series between a pair of sensors to create an edge and find similarity even when the values of the two time-series vary. Therefore, we do not consider P_{corr} and P_{deconv} henceforth. On the basis of our observations, we find that P_{nei} can be used irrespective of the dataset and E_d, P_{haar} can be used only for datasets with small number of sensors or sensors when E_d is greater than 0.40 while P_{dtw} can be used for small networks. P_{gsp} can be used only if the threshold is tuned for different edge densities.

Total cumulative energy residual

We compare the $TCER$ value of a graph to that of a random graph for a dataset. Our observations indicate that $\mathcal{G}_{dtw}$ and $\mathcal{G}_{haar}$ always yield the best $TCER$ values, around $0.88 - 0.98$ irrespective of the E_d and the dataset and $\mathcal{G}_{nei}$ has the lowest $TCER$ values. We show our observations in Table B.4. We observe that the $TCER$ for D_{ws} is bad irrespective of the graph creation approach. The reason for this is that the $TCER$ value for the original graph from which we simulate the data for D_{ws} is much lower than 1, i.e., 0.89.

Reconstruction Error

We compare the reconstruction error of all the graph creation approaches on D_{epa}-D_{ws} for when E_d are 0.20, 0.40, 0.60 and 0.75 and the number of *representative sampling subsets* are 5, 7, 10 and 13 respectively. We perform this experiment to understand how the choice of the graph creation approach affects the reconstruction error. For our experiments, we select *Frob* here. We perform this experiment only for connected graphs. Our observations indicate that P_{corr} has significantly higher reconstruction error than others whereas P_{nei}, P_{haar} and P_{dtw} have similar results for D_{epa}. We observe P_{nei} has a higher reconstruction error than P_{haar}, P_{gsp} and P_{dtw} for D_{temp} and D_{pol}. Furthermore, P_{dtw} and P_{haar} performs the best followed by P_{gsp} irrespective of the dataset and E_d. We show the results for D_{epa} in Figure B.3 where we observe that P_{haar} performs the best followed by P_{dtw} while P_{nei} and P_{corr} performs the worst.

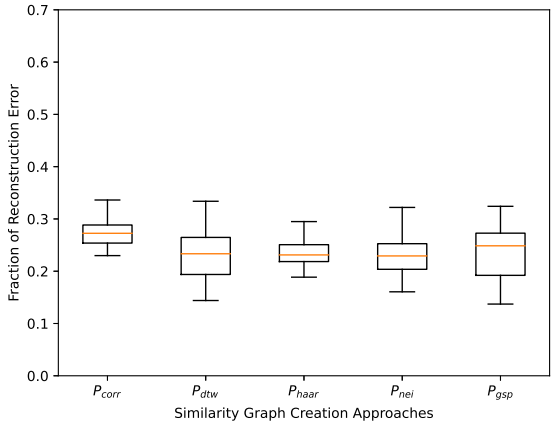


Fig. B.3: Comparison of Reconstruction Error when E_d is 0.75 of P_{corr}, P_{dtw}, P_{haar}, P_{nei} and P_{gsp} for D_{epa}. The x-axis represents the similarity graph creation approaches of Phase-I and y-axis represents the reconstruction error.

Summary for Phase-I

We conclude that P_{haar} followed by P_{dtw} is the best choice for Phase-I for a dataset with more than 90 nodes and high E_d (more than 0.40) and for any E_d for a dataset with less than 90. However, for graphs with more than 90 nodes and E_d less than 0.40, we recommend P_{nei} followed by P_{gsp}. We use these observations to propose *AutoSub-GraphSample*. As already discussed, we do not recommend P_{corr} and P_{deconv}.

7.2 Phase-II Results: Comparison of the Sampling Techniques

We compare the performance of the sampling approaches for Phase-II and a random *representative sampling subset* selection algorithm on the basis of their solution for Equation B.2. For our experiments, we compare the reconstruction error generated by the sampling techniques for each graph creation approach with different E_d, such as, 0.20, 0.40, 0.60 and 0.75 and vary K from $5-13$ and calculate the *reconstruction error* quartile.

We find that irrespective of K and the dataset, *SRel* ranks $1-3$ among all sampling techniques when the E_d is greater than 0.40 whereas *SMMR* ranks $1-3$ when the E_d is less than 0.40. *SEMMR* has similar mean *reconstruction error* as *SMMR* but, the maximum *reconstruction error* is much higher. *SRel* has around $0.1-0.40$ reconstruction error when E_d is greater than 0.40 and $0.20-0.60$ otherwise. *SMMR* and *SEMMR* has around $0.2-0.40$ when E_d is less than 0.40 and $0.10-0.60$ otherwise. *SRel*, *SMMR* and

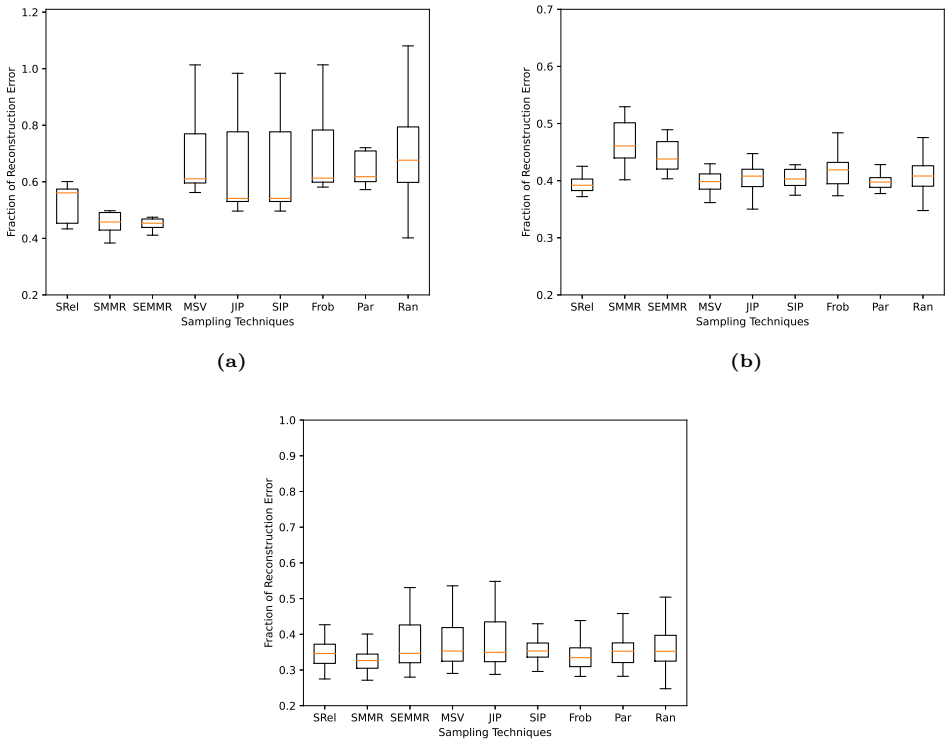


Fig. B.4: Comparing the reconstruction error of sampling techniques and Random Sampling Approach for K is 5, E_d is 0.20 on $\mathcal{G}_{nei}$ in Figure B.4a of D_{temp}, for K is 7 when E_d is 0.75 on $\mathcal{G}_{haar}$ in Figure B.4b of D_{temp} and K is 5, E_d is 0.75 on $\mathcal{G}_{haar}$ in Figure B.4c of D_{pol}.

SEMMR has the highest maximum reconstruction error for D_{epa} when E_d is greater than 0.60 and is K greater than 7. *MSV* has around $0.20 - 0.40$ reconstruction error when E_d is around 0.75 but the performance degrades for low E_d to around $0.30 - 0.80$ reconstruction error. *MSV* also has the highest maximum reconstruction error at low E_d. On comparing *JIP* and *SIP* which follow similar approaches, we observe that *SIP* yields better performance than *JIP* in every scenario irrespective of K, E_d or dataset. On comparison with the other sampling approaches, we observe that *SIP* has around $0.40 - 0.60$ reconstruction error when E_d is high and $0.40 - 0.90$ otherwise. *Par* has the worst performance among all sampling techniques. Although *Frob* ranks in the top $3 - 4$ among all sampling techniques based on the minimum *reconstruction error*, it produces the maximum reconstruction error among all sampling techniques. As expected, we also

observe that the reconstruction error increases with increase in K irrespective of the sampling technique.

Based on our observations, we conclude that *SRel* is the best choice for graphs with high E_d (greater than 0.40) and *SMMR* for graphs with E_d less than 0.40. However, if we need to choose sampling technique that performs irrespective of the E_d, *Frob* should be selected. We use these observations to propose *AutoSubGraphSample*. Due to the huge number of results, we only show 3 representative examples in Figure B.4.

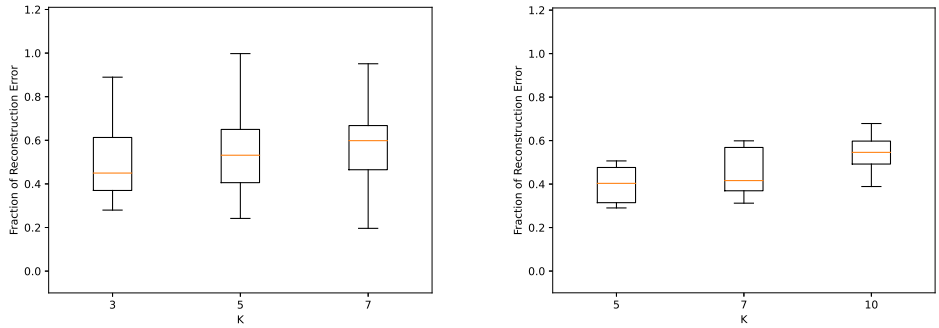
7.3 Evaluation of AutoSubGraphSample

Based on our observations for Phase-I and Phase-II, we decide the values for Th_n and Th_e in Algorithm *AutoSubGraphSample* as 90 and 0.40 respectively. We analyze the generalizability of *AutoSubGraphSample* on 9 new representative datasets now.

1. D_{ps}: A dataset that records temperature of 55 sensors.[3]
2. D_{in}: A dataset that records humidity of 54 sensors.[4]
3. D_{hum}: A dataset that records humidity of 100 sensors.[5]
4. D_{gas}: A dataset that records acetone of 16 sensors.[6]
5. D_{sof}: A dataset that records temperature of 170 sensors.[7]
6. D_{elec}: A dataset that records the electricity consumption of 124 sensors.[8]
7. D_{pre}: A dataset that records pressure of 324 sensors.[9]
8. D_{syn}: We generate synthetic dataset that records temperature of 1000 sensors
9. D_{syn1}: We generate synthetic dataset that records the humidity consumption of 1000 sensors.

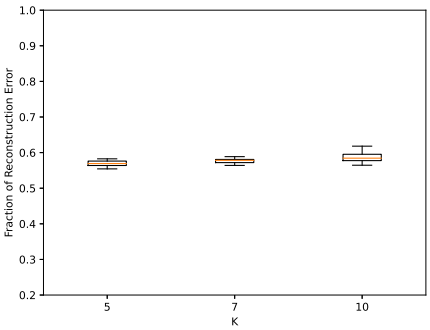
Based on *AutoSubGraphSample*, we apply P_{haar} in Phase-I irrespective of the E_d and *SMMR* or *SRel* in Phase-II on the basis of E_d for D_{ps}, D_{in} and D_{gas}. However, for D_{hum}, D_{sof}, D_{syn}, D_{syn1} and D_{elec}, we apply P_{haar} in Phase-I and *SRel* in Phase-II when E_d is greater than 0.40 and P_{nei} in Phase-I followed by *SMMR* in Phase-II otherwise. We consider the number of *representative sampling subsets*, K as 5, 7 and

[3] https://archive.ics.uci.edu/ml/datasets.php
[4] https://www.kaggle.com/hmavrodiev/air-quality-dataset?select=2017-09_bme280sof.csv
[5] https://archive.ics.uci.edu/ml/datasets/
[6] https://archive.ics.uci.edu/ml/datasets/
[7] https://www.kaggle.com/hmavrodiev/sofia-air-quality-dataset
[8] https://archive.ics.uci.edu/ml/datasets/
[9] https://www.kaggle.com/code/wessam1234/9394b134-a

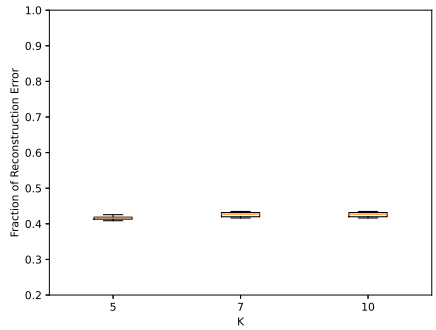


(a)

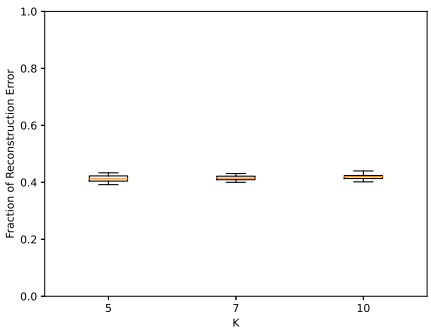
(b)



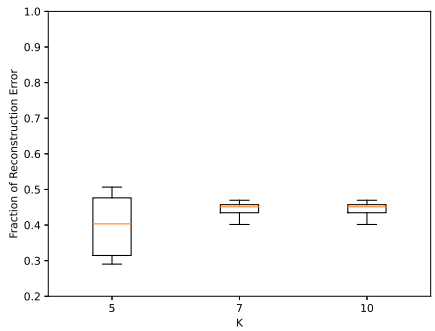
(c)



(d)



(e)



(f)

Fig. B.5: Comparing the reconstruction error of D_{gas} in Figure B.5a for $E_d = 0.25$, D_{hum} in Figure B.5b for $E_d = 0.25$, D_{pre} in Figure B.5c for $E_d = 0.60$, D_{syn} in Figure B.5d for $E_d = 0.60$, D_{sof} in Figure B.5e for $E_d = 0.60$ and D_{syn1} in Figure B.5f for $E_d = 0.75$. The x-axis represents K and y-axis represents the reconstruction error.

Dataset	E_d	*AutoSubGraphSample*	*Manual*
D_{ps}	0.25	0.51	0.51
	0.75	0.61	0.61
D_{hum}	0.25	0.31	0.31
	0.75	0.42	0.42
D_{sof}	0.25	0.41	0.41
	0.75	0.38	0.38
D_{gas}	0.25	0.49	0.47
	0.75	0.56	0.56
D_{in}	0.25	0.33	0.32
	0.75	0.49	0.49
D_{elec}	0.25	0.47	0.45
	0.75	0.50	0.47

Table B.5: Comparing the reconstruction error by *AutoSubGraphSample* and *Manual Selection* on D_{ps}, D_{gas}, D_{in}, D_{elec}, D_{sof} and D_{hum} when E_d is 0.25 and 0.75 for K = 5.

10 for all datasets except D_{gas}. As D_{gas} comprises of only 16 sensors, we consider K as 3, 5 and 7. We observe that the reconstruction error by *AutoSubGraphSample* is similar for all datasets except for D_{gas} and $D_{elec}$to our previous observations for other datasets irrespective of the number of *representative sampling subsets* and E_d. Our observations indicate that the reconstruction error for D_{elec} is high when K increases more than 5, therefore we can increase the battery life for D_{elec} by maximum 5 times within the error margin of 0.40. We observe for D_{gas} the highest reconstruction error is greater than 0.60 whereas the average is within 0.40 for K values ranging from $3-7$. The reason being the number of sensors being very less, i.e., only 16. We, further, observe that *AutoSubGraphSample* can ensure similar performance irrespective of the number of sensors as the number of sensors in the datasets range from $16-1000$. We show some representative examples of our observations in Figure B.5. In order to understand the significance of *AutoSubGraphSample*, we compare the performance by *AutoSubGraphSample* and *Manual Selection*, i.e., if we manually select the best combination of Phase-I and Phase-II algorithms specifically for a dataset. We apply all combinations of Phase-I and Phase-II algorithms on a dataset and calculate the respective reconstruction errors for a specific E_d. We select that combination of Phase-I and Phase-II algorithm which provides the least reconstruction error as the *Manual Selection*. We repeat this for all the 9 datasets when E_d is $0.25-0.75$ and K as 3, 5, 7 and 10. We show our observations as shown in Table B.5 for K = 5, E_d is 0.25 and 0.75 shows that *AutoSubGraphSample* can ensure similar results as compared to *Manual Selection* with a small margin of around $2-6\%$ for D_{gas} and D_{elec} and same results for other datasets. Therefore, based on our observations, we can conclude that

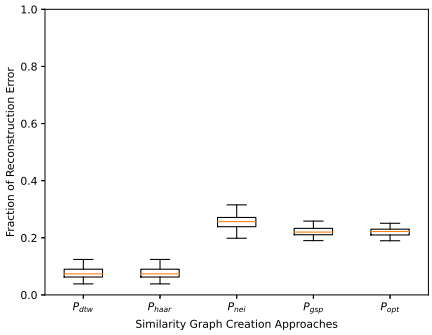


Fig. B.6: Comparing the reconstruction error for the optimum sampling algorithms on $\mathcal{G}_{nei}$, $\mathcal{G}_{haar}$, $\mathcal{G}_{dtw}$, $\mathcal{G}_{gsp}$, and $\mathcal{G}_{opt}$ on D_{ex} is shown.

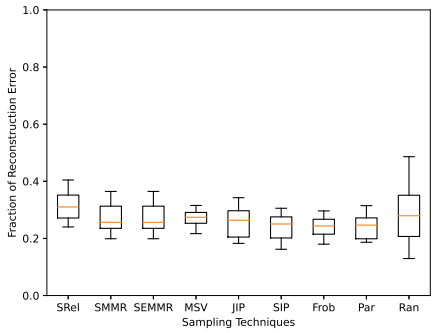


Fig. B.7: Comparing the reconstruction error of the sampling techniques on $\mathcal{G}_{opt}$ for D_{ex} is shown.

AutoSubGraphSample generalizes to a dataset irrespective of the size of the dataset and E_d.

7.4 Comparing SubGraphSample with Exhaustive Search

In theory, identification of the *optimum representative sampling subsets* given the data of the sensors is possible through a joint exhaustive search for both the best graph topology and best sampling partition. However, this requires us to perform an exhaustive search for best sampling partition for every possible graph topology which is so computationally expensive that we consider it to be infeasible. Therefore, we consider this in 2 phases, where in Phase-I, we search for an *optimum graph topology* and in Phase-II, we search for the *optimum sampling partition* given the *optimum graph topology*. For our experiments, we consider a subset of 8 sensors of D_{pol}, namely D_{ex}, as exhaustive analysis is not possible on the complete dataset.

Method	E_d	**Avg** P_l	**Avg** CC	**TCER**
$\mathcal{G}_{opt}$	0.64	1.36	0.45	0.88
$\mathcal{G}_{nei}$	0.46	1.54	0.82	0.82
$\mathcal{G}_{haar}$	0.50	1.79	f0.87	**0.86**
$\mathcal{G}_{dtw}$	0.50	1.79	0.87	**0.86**
$\mathcal{G}_{gsp}$	0.57	1.54	0.80	0.83

Table B.6: E_d, P_l, CC and TCER for D_{ex} is shown

In order to identify the *optimum graph*, we explore the relationship between existing

graph topology measures, like *average path length*, *clustering coefficient* and *TCER* with *optimal graph*. Based on our observations, we conclude that the *TCER* values are indirectly proportional to *reconstruction error*, i.e., higher the *TCER* values, lower is the *reconstruction error*. Furthermore, if different graphs have similar *TCER* values, we observe that as the *average path length* decreases, the *reconstruction error* also decreases. We calculate the *TCER* for all possible connected graphs to a precision of 2 significant digits for D_{ex}. We consider the graph which has the highest *TCER* and shortest *average path length* as the *optimal graph*, $\mathcal{G}_{opt}$. As the E_d of $\mathcal{G}_{opt}$ is 0.64, we try to find graphs with similar E_d using the proposed methods. We show the E_d, average path length, clustering co-efficient, *TCER* of $\mathcal{G}_{opt}$ with $\mathcal{G}_{nei}$, $\mathcal{G}_{haar}$, $\mathcal{G}_{dtw}$ and $\mathcal{G}_{gsp}$ in Table B.6 which indicates that $\mathcal{G}_{haar}$ and $\mathcal{G}_{dtw}$ has the most similar *TCER* values with $\mathcal{G}_{opt}$. As the graphs produced with P_{haar} and P_{dtw} are identical, so we only show results for P_{dtw} henceforth.

In order to find the O_s, we search all possible sampling partitions on $\mathcal{G}_{opt}$ such that the maximum *reconstruction error* is the lowest. We perform an exhaustive search to find O_s on $\mathcal{G}_{nei}$, $\mathcal{G}_{dtw}$, $\mathcal{G}_{gsp}$ and $\mathcal{G}_{opt}$. Our observations as shown in Figure B.6 indicate that P_{haar} and P_{dtw} ensures the least reconstruction error. Therefore, our observations indicate that it is possible to find a graph that gives lower reconstruction error than the graph with the highest *TCER*. To evaluate the different sampling algorithms for Phase-II, we compare the reconstruction error by *Frob*, *MSV*, *SIP*, *SMMR* and *SRel* on $\mathcal{G}_{opt}$ in Figure B.7. Our observations indicate that by *Frob*, *SIP* and *SMMR* has the least reconstruction error with respect to *Opt*. However, these observations varies with network size and edge density. As it is not possible to confirm every scenario of different edge densities and for different network sizes with exhaustive analysis, we compare the performance of the graph creation approaches and sampling algorithms on a synthetic dataset whose *representative sampling subsets* are already provided next in Subsection 7.5.

7.5 Comparison of SubGraphSample with *optimum Sampling Sets*

We now evaluate how close the *representative sampling subsets* found by *SubGraphSample* are to the optimum sampling sets, O_s. As we do not have O_s for any real dataset, we construct a dataset such that we know O_s. We assume the optimal number of sampling sets, K as 6, the total number of sensors, N as 40, the length of the time-series as 10, sensors as $S_1, S_2, \ldots, S_N$ and we denote this dataset as D_{st}. We simulate D_{st} such that it records temperature. We generate O_s of 6, $O_1, O_2, \ldots, O_6$ sampling sets by randomly allocating each sensor to a O_i on the basis of D_{st}. Based on O_s, we generate the time-series of S such that while the mean values of the distributions vary by $3-5$ between different sampling sets, i.e., the constructed sampling sets are indeed the optimal.

We calculate the reconstruction error of O_s for D_{st} to understand the performance

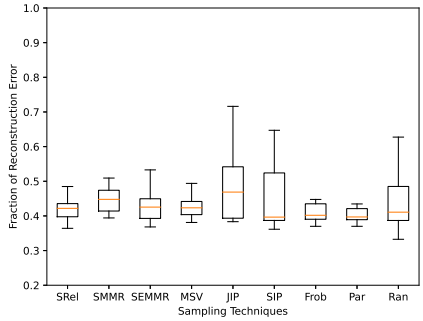


Fig. B.8: Comparison of the reconstruction error of when E_d is 0.60 and P_{haar} in Phase-I for D_{st} is shown.

Phase-I	E_d	**Avg** P_l	**Avg** CC	**TCER**	**Th**
$\mathcal{G}_{dtw}$	0.23	inf	0.91	0.49	120
	0.40	2.26	0.84	0.97	210
	0.60	1.47	0.89	0.97	330
$\mathcal{G}_{nei}$	0.23	1.78	0.89	0.89	5
	0.40	1.60	0.86	0.85	9
	0.60	1.39	0.84	0.87	15
$\mathcal{G}_{haar}$	0.23	2.64	0.81	0.97	60
	0.40	1.77	0.77	0.96	100
	0.61	1.39	0.81	0.93	185
$\mathcal{G}_{gsp}$	0.20	2.05	0.39	0.90	0.01
	0.40	1.69	0.52	0.89	-4e-06
	0.59	1.47	0.71	0.89	-1.1e-05

Table B.7: Average path length (P_l), clustering co-efficient (CC), TCER and the threshold value for D_{st} is shown

of O_s. As we do not know the true E_d and the graph topology of D_{st} which is required to calculate the reconstruction error, we consider 4 different E_d, such as, 0.20, 0.40, 0.60 and 0.75 and the 4 similarity graph creation algorithms, P_{dtw}, P_{haar}, P_{nei} and P_{gsp}. We compare P_{dtw}, P_{nei}, P_{gsp} and P_{haar} on the basis graph topology, $TCER$ and the reconstruction error for all E_d in Table B.7. Our observations shows that O_s has minimum reconstruction error when E_d is 0.60 and similarity graph creation approach is P_{haar}. On comparing the sampling techniques on $\mathcal{G}_{haar}$ when E_d is 0.60, our observations as shown in Figure B.8 indicate that *SRel* produces similar reconstruction

error to O_s. Therefore, the combination of P_{haar} and *SRel* can ensure most similar results to O_s. On the basis of our observations from Subsection 7.5 and this Subsection, we find that the proposed recommendations for Algorithm *AutoSubGraphSample* can ensure most similar results to O_s. For example, we observe that P_{haar} in Phase-I, *SMMR* or *Frob* in Phase-II has the best performance. Although it is not possible to confirm every scenario by exhaustive analysis, our results from empirical analysis supports the recommendations by Algorithm *AutoSubGraphSample* when P_{nei} is used in Phase-I and when *SRel* could be used in Phase-II.

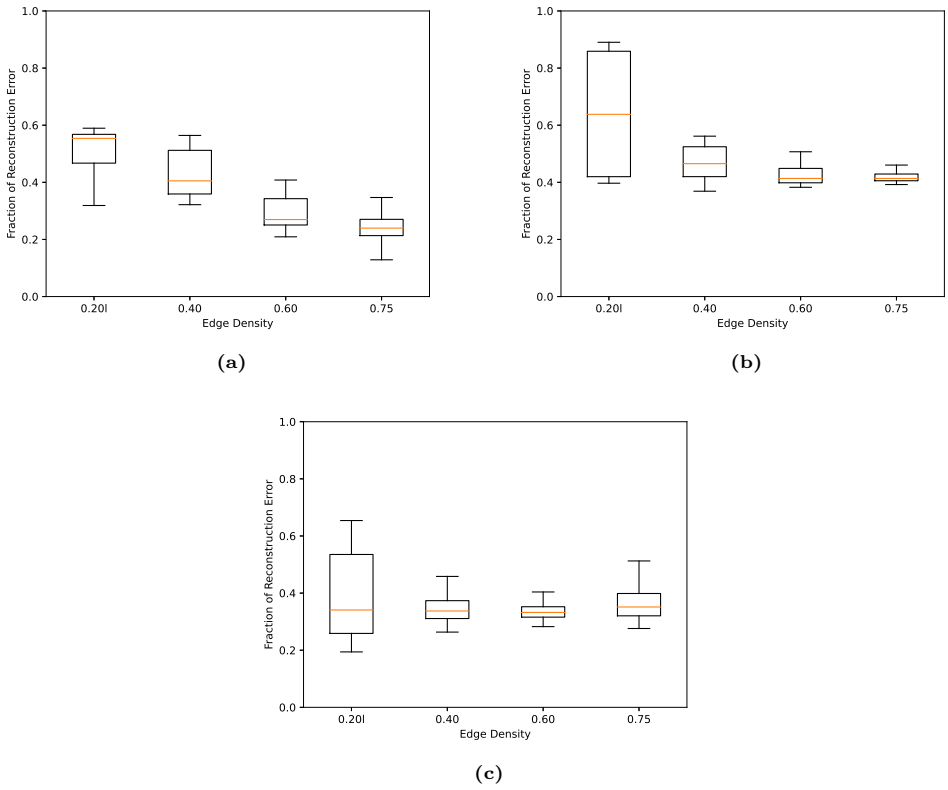


Fig. B.9: Comparison of the reconstruction error for different edge densities for $\mathcal{G}_{gsp}$ on D_{epa} in Figure B.9a, $\mathcal{G}_{nei}$ on D_{temp} in Figure B.9b and $\mathcal{G}_{dtw}$ on B.9c in Figure 9c is shown. The x-axis represents the sampling techniques and y-axis represents the reconstruction error.

7.6 Studying the impact of E_d on Reconstruction Error

To study the relationship between E_d and reconstruction error, we calculate the reconstruction error for different edge densities $0.20 - 0.75$. For this experiment, we consider *Frob* for Phase-II and K as 7. Our observations differ with respect to datasets. For example, we observe that the higher the E_d, the lower is the reconstruction error for D_{epa} as shown in Figure B.9a. However, we observe that the reconstruction error is the highest when E_d is 0.2 and decreases with increase in E_d for D_{temp} as shown in Figure B.9b. We did not observe any relationship between E_d and reconstruction error in D_{pol} and D_{ws} as shown in Figure B.9c. Based on these observations, we conclude there is an optimal E_d for which the reconstruction error is the lowest for a dataset. However, the optimal E_d differs across datasets.

7.7 Frequency analysis

In order to understand the performance of the graph creation approaches discussed in Phase-I, we visualize the Frequency transform of the graphs created given a dataset. As discussed in Section 3.2, Graph Fourier Transform (GFT) is the eigen decomposition of the graph Laplacian, L into eigenvalues, Λ and eigenvectors, V, i.e., GFT of L is $L = V\Lambda V^{-1}$. Additionally, GFT of the graph signal at the $k-$th time-stamp, x^k, is $\hat{x}^k$ which is defined as $\hat{x}^k = V^{-1}x^k$. Therefore, given a dataset, the GFT of the optimal graph topology should comprise of the maximum number of possible distinct eigenvalues which are evenly spread. Additionally, the GFT of the optimal graph topology should be such that the lower the eigenvalues, the higher the amplitudes and vice-versa. Our observations indicate that the GFT of $\mathcal{G}_{haar}$ and $\mathcal{G}_{dtw}$ ensures optimal graph topology created given a dataset and the E_d whereas $\mathcal{G}_{nei}$ has fewer distinct eigenvalues and therefore, is not optimal for D_{pol} and D_{temp}. We show some representative examples of our observations in Figure B.10.

7.8 Evaluation of *AutoSubGraphSample* on partial Time Series

In our previous experiments, we generate $\mathcal{G}$ based on the complete time-series given a dataset. In this Subsection, we analyze the performance of *AutoSubGraphSample* when the complete time-series is not available. Therefore, in this experiment, we select a fraction, p, of the time-series for which to generate the similarity graph, $\mathcal{G}'$, and identify K sampling sets on $\mathcal{G}'$. We compare the reconstruction error of $\mathcal{G}'$ with $\mathcal{G}$ for different values of p while keeping K, E_d, Phase-I and Phase-II approaches constant. We repeat this experiment by varying p as 0.75, 0.50 and 0.25 of the time-series, K between $5 - 13$ and E_d between $0.20 - 0.75$, respectively. We repeat this for all the 13 datasets. Our observations as shown in Figure 11 indicate that the difference in average reconstruction error is minimal $(1 - 9\%)$ across datasets irrespective of p. Therefore, we

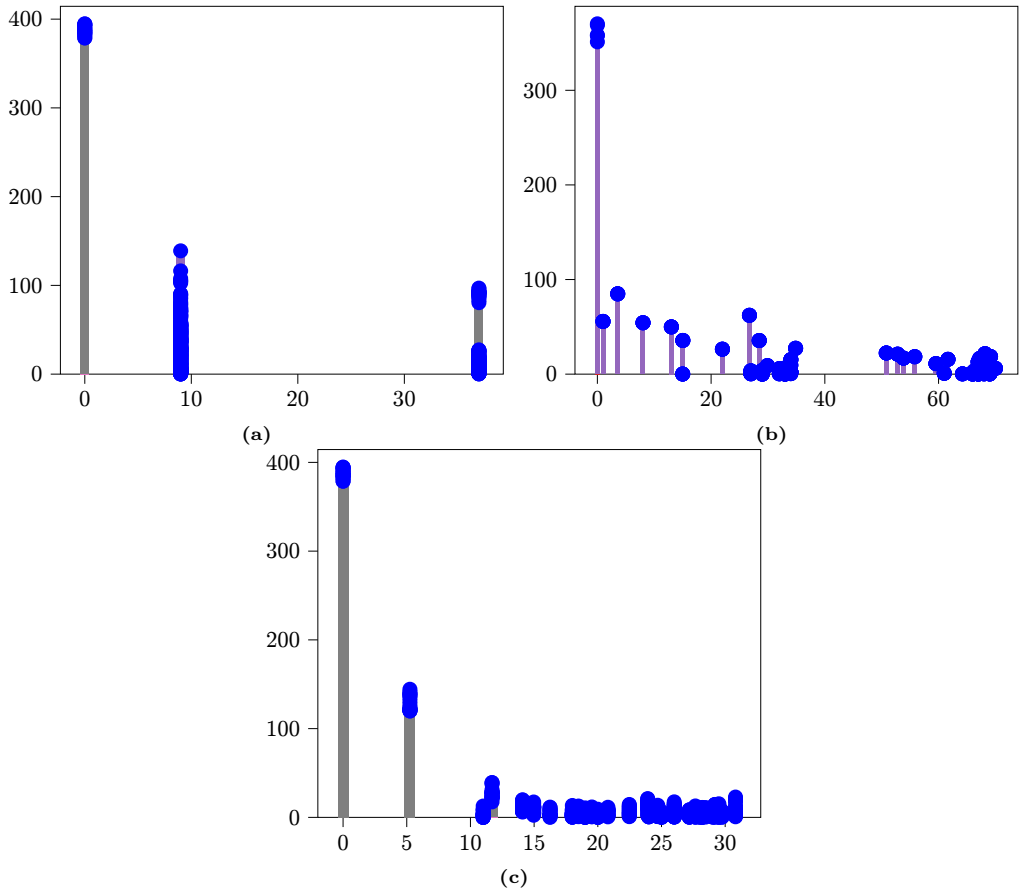


Fig. B.10: Graph Fourier transforms of $\mathcal{G}_{nei}$ with E_d 0.43 on D_{pol} in Figure 10a, $\mathcal{G}_{haar}$ with E_d 0.40 on D_{temp} in Figure 10b and $\mathcal{G}_{dtw}$ with E_d 0.60 on D_{pol} in Figure 10c is shown

can conclude that *AutoSubGraphSample* can ensure similar increase in battery longevity within the error bound irrespective of the size of the time-series. We intuitively believe the reason being *AutoSubGraphSample* identifies a list of sampling sets such that similar sensors are allocated to different sampling sets by utilizing the similarity graph topology. Therefore, although the sampling sets might vary as we vary p of the time-series, it does not impact the reconstruction error. Furthermore, there is generally a correlation in the sensor's data irrespective of the p which is another reason that the performance of *AutoSubGraphSample* is not impacted and *AutoSubGraphSample* can handle small fluctuations in the sensing data easily by the utilization of similarity graph topology.

7.9 Identifying the Maximum Number of Sampling Sub-sets, K

In this Subsection, we compare the performance of Phase-II for Equation B.1, i.e., we find the K generated by a sampling approach given ε. However, the proposed sampling approaches except *SRel*, *SMMR* and *SEMMR* can not provide a solution for Equation B.1 as they focus on identifying the maximum error given K and therefore, they require K to be pre-specified and can not be modified to identify the maximum number of sampling sub-sets given ε. Therefore, we select *SRel*, *SMMR* and *SEMMR* for this experiment. We compare K generated by *SRel*, *SMMR* and *SEMMR* for each graph creation approach with different E_d, such as, $0.20 - 0.75$, and vary ε from $0.40 - 0.60$. Our observations indicate that *SRel* generates the maximum value for K for high E_d whereas *SMMR* generates the maximum value for low E_d. Although *SEMMR* performs similar to *SMMR* for all datasets, the performance for *SEMMR* is the worst among all for D_{pol} irrespective of E_d. We show our observations on $\mathcal{G}_{dtw}$, $\mathcal{G}_{nei}$, and $\mathcal{G}_{haar}$ for E_d as 0.20, 0.40, 0.75 on D_{pol} and D_{temp} datasets for ε as 0.40 and 0.50 respectively in Table B.8.

8 Conclusions and Future Works

In this paper, we propose *SubGraphSample* which finds the maximum number of *representative sampling subsets* given a *sensor graph*. By finding the maximum number of *representative sampling subsets*, we can alternate querying between these and thus, increase battery longevity significantly. Unlike existing sampling approaches, *SubGraphSample* do not require prior knowledge of the similarity of the sensors and automatically identifies the maximum number of *representative sampling subsets*. We explore 6 graph creation approaches and propose 6 sampling approaches in *SubGraphSample*. However, the suitability and performance of a graph creation approach and sampling approach varies across datasets. Therefore, we propose Algorithm *AutoSubGraphSample* which can autoselect the most suitable approaches given a *sensor graph* and we, further, show the generalizability of *AutoSubGraphSample* given a dataset. We evaluate all possible combination of approaches of *SubGraphSample* on 4 datasets which shows that the best

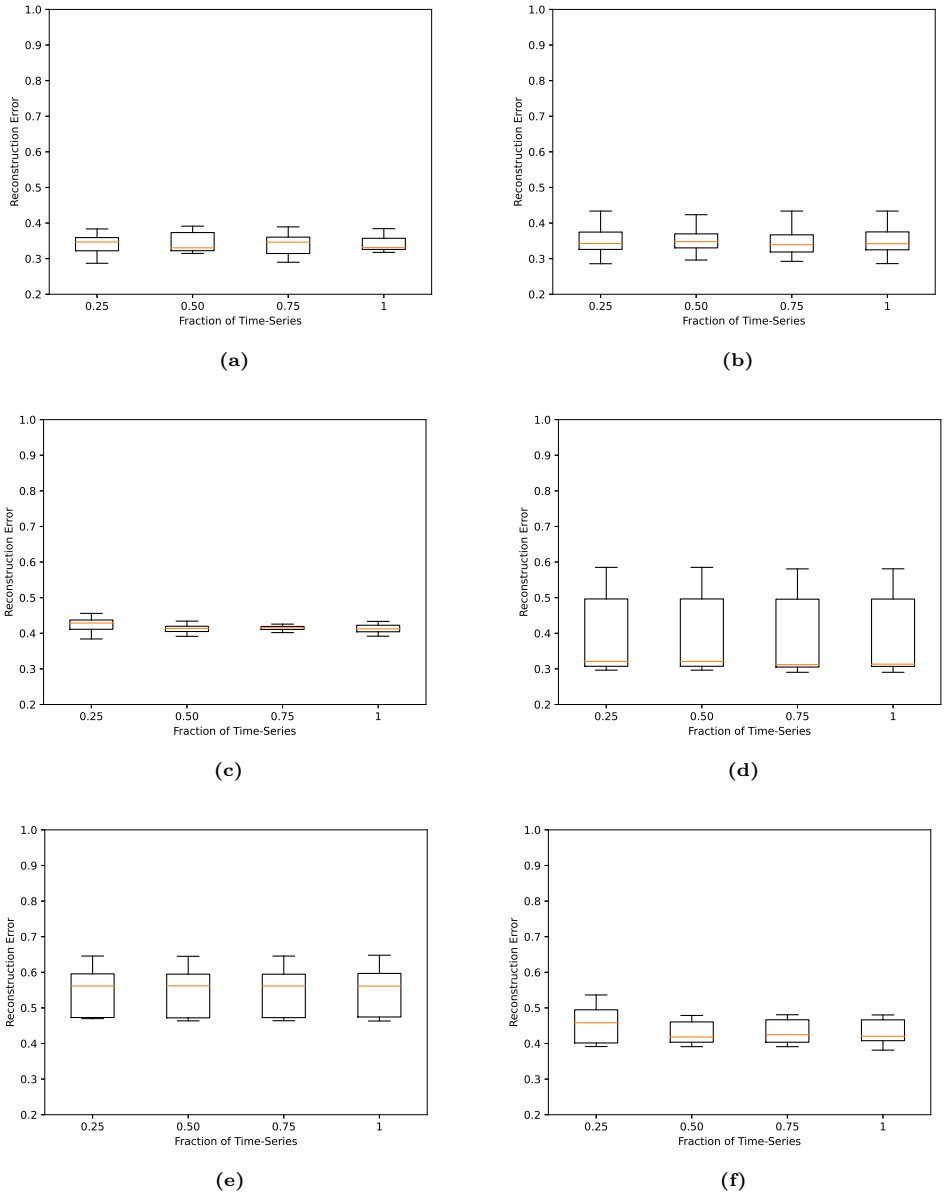


Fig. B.11: Comparing the reconstruction error of D_{temp} when K = 7, $E_d = 0.75$ in Figure B.11a, D_{pol} when K = 5, $E_d = 0.60$ in Figure B.11b, D_{sof} when K = 5, $E_d = 0.60$ in Figure B.11c, D_{hum} when K = 7, $E_d = 0.40$ in Figure B.11d, D_{ps} when $E_d = 0.75$, when K = 5 in Figure B.11e and D_{temp} when K = 5, $E_d = 0.20$ in Figure B.11f. The x-axis represents the fraction of time-series, p and y-axis represents the reconstruction error.

Dataset	ε	E_d	**Phase-I**	**Phase-II**	K
D_{temp}	0.40	0.75	$\mathcal{G}_{nei}$	*SRel*	6
				SMMR	4
				SEMMR	4
D_{temp}	0.40	0.20	$\mathcal{G}_{nei}$	*SRel*	3
				SMMR	4
				SEMMR	4
D_{temp}	0.40	0.75	$\mathcal{G}_{haar}$	*SRel*	5
				SMMR	4
				SEMMR	4
D_{temp}	0.40	0.40	$\mathcal{G}_{haar}$	*SRel*	9
				SMMR	10
				SEMMR	10
D_{pol}	0.50	0.75	$\mathcal{G}_{nei}$	*SRel*	7
				SMMR	7
				SEMMR	4
D_{pol}	0.50	0.20	$\mathcal{G}_{nei}$	*SRel*	3
				SMMR	4
				SEMMR	3
D_{pol}	0.50	0.75	$\mathcal{G}_{dtw}$	*SRel*	8
				SMMR	7
				SEMMR	5
D_{pol}	0.50	0.20	$\mathcal{G}_{dtw}$	*SRel*	4
				SMMR	4
				SEMMR	3

Table B.8: We show the number of sampling sets, K generated by *SRel*, *SMMR* and *SEMMR* when the mean reconstruction error, ε, is given for D_{temp} and D_{pol}

combination of algorithms can provide $5 - 13$ times increase in battery life within a $20 - 40\%$ error bound.

As a future work, we will extend *AutoSubGraphSample* to handle multivariate time-series and scale to large time-series using deep learning-based time-series embedding. Furthermore, we aim to merge the current two phases into one in a deep reinforcement learning based model.

Acknowledgment

This work has, in part, been supported by the Danish Council for Independent Research (Grant No. 8022-00284B SEMIOTIC).

References

[1] S. Ashraf, S. Saleem, and T. Ahmed, "Sagacious communication link selection mechanism for underwater wireless sensors network," *Int. J. Wirel. Microw. Technol*, vol. 10, no. 2, pp. 12–25, 2020.

[2] Y.-B. Chen, I. Nevat, P. Zhang, S. G. Nagarajan, and H.-Y. Wei, "Query-based sensors selection for collaborative wireless sensor networks with stochastic energy harvesting," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3031–3043, 2018.

[3] J. Paparrizos, C. Liu, B. Barbarioli, J. Hwang, I. Edian, A. J. Elmore, M. J. Franklin, and S. Krishnan, "Vergedb: A database for iot analytics on edge devices." in *CIDR*, 2021.

[4] D. Lin, Q. Wang, W. Min, J. Xu, and Z. Zhang, "A survey on energy-efficient strategies in static wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 17, no. 1, pp. 1–48, 2020.

[5] T. V. D. Lee, G. Exarchakos, and S. H. D. Groot, "Distributed reliable and energy-efficient scheduling for lr-wpans," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 4, pp. 1–20, 2020.

[6] L. Mao and L. Jackson, "Selection of optimal sensors for predicting performance of polymer electrolyte membrane fuel cell," *Journal of Power Sources*, vol. 328, pp. 151–160, 2016.

[7] H. J. Diamond, T. R. Karl, M. A. Palecki, C. B. Baker, J. E. Bell, R. D. Leeper, D. R. Easterling, J. H. Lawrimore, T. P. Meyers, M. R. Helfert *et al.*, "Us climate reference network after one decade of operations: Status and assessment," *Bulletin of the American Meteorological Society*, vol. 94, no. 4, pp. 485–498, 2013.

[8] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[9] Y. Tanaka, Y. C. Eldar, A. Ortega, and G. Cheung, "Sampling signals on graphs: From theory to applications," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 14–30, 2020.

[10] N. Perraudin, B. Ricaud, D. I. Shuman, and P. Vandergheynst, "Global and local uncertainty principles for signals on graphs," *APSIPA Transactions on Signal and Information Processing*, vol. 7, 2018.

[11] L. F. Chamon and A. Ribeiro, "Greedy sampling of graph signals," *IEEE Transactions on Signal Processing*, vol. 66, no. 1, pp. 34–47, 2017.

[12] A. Gadde, A. Anis, and A. Ortega, "Active semi-supervised learning using sampling theory for graph signals," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 492–501.

[13] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[14] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under laplacian and structural constraints," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 825–841, 2017.

[15] S. Hassan-Moghaddam, N. K. Dhingra, and M. R. Jovanović, "Topology identification of undirected consensus networks via sparse inverse covariance estimation," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 4624–4629.

[16] S. Feizi, D. Marbach, M. Médard, and M. Kellis, "Network deconvolution as a general method to distinguish direct dependencies in networks," *Nature biotechnology*, vol. 31, no. 8, pp. 726–733, 2013.

[17] W. Jiang, "Time series classification: Nearest neighbor versus deep learning models," *SN Applied Sciences*, vol. 2, no. 4, pp. 1–17, 2020.

[18] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[19] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 2004, pp. 792–803.

[20] B.-K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary lp norms," *KiltHub*, 2000.

[21] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *ACM Sigmod Record*, vol. 23, no. 2, pp. 419–429, 1994.

[22] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*. Seattle, WA, USA:, 1994, pp. 359–370.

[23] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th international conference on data engineering*. IEEE, 2002, pp. 673–684.

[24] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 491–502.

[25] M. D. Morse and J. M. Patel, "An efficient and accurate method for evaluating time series similarity," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, 2007, pp. 569–580.

[26] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. Tung, "Spade: On shape-based pattern detection in streaming time series," in *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 2007, pp. 786–795.

[27] M. Cuturi, "Fast global alignment kernels," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 929–936.

[28] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *International Conference on Machine Learning*. PMLR, 2017, pp. 894–903.

[29] J. Yin, R. Wang, H. Zheng, Y. Yang, Y. Li, and M. Xu, "A new time series similarity measurement method based on the morphological pattern and symbolic aggregate approximation," *IEEE Access*, vol. 7, pp. 109 751–109 762, 2019.

[30] V. Stojov, N. Koteli, P. Lameski, and E. Zdravevski, "Application of machine learning and time-series analysis for air pollution prediction," *Proceedings of the CIIT*, 2018.

[31] T. Jaakkola, M. Diekhans, and D. Haussler, "A discriminative framework for detecting remote protein homologies," *Journal of computational biology*, vol. 7, no. 1-2, pp. 95–114, 2000.

[32] W. Pei, H. Dibeklioğlu, D. M. Tax, and L. van der Maaten, "Multivariate time-series classification using the hidden-unit logistic model," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 4, pp. 920–931, 2017.

[33] E. Zdravevski, P. Lameski, R. Mingov, A. Kulakov, and D. Gjorgjevikj, "Robust histogram-based feature engineering of time series data," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2015, pp. 381–388.

[34] E. Zdravevski, P. Lameski, V. Trajkovik, A. Kulakov, I. Chorbev, R. Goleva, N. Pombo, and N. Garcia, "Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering," *Ieee Access*, vol. 5, pp. 5262–5280, 2017.

[35] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM Sigkdd Explorations Newsletter*, vol. 12, no. 1, pp. 40–48, 2010.

[36] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[37] Z. Gong and H. Chen, "Sequential data classification by dynamic state warping," *Knowledge and Information Systems*, vol. 57, no. 3, pp. 545–570, 2018.

[38] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, 2015.

[39] A. Abid and J. Zou, "Autowarp: Learning a warping distance from unlabeled time series using sequence autoencoders," *arXiv preprint arXiv:1810.10107*, 2018.

[40] S. Matsuo, X. Wu, G. Atarsaikhan, A. Kimura, K. Kashino, B. K. Iwana, and S. Uchida, "Attention to warp: Deep metric learning for multivariate time series," *arXiv preprint arXiv:2103.15074*, 2021.

[41] J. Narwariya, P. Malhotra, L. Vig, G. Shroff, and T. Vishnu, "Meta-learning for few-shot time series classification," in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*. ACM, 2020, pp. 28–36.

[42] D. Yao, G. Cong, C. Zhang, X. Meng, R. Duan, and J. Bi, "A linear time approach to computing time series similarity based on deep metric learning," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[43] A. Venkitaraman, S. Chatterjee, and P. Händel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 698–710, 2019.

[44] T. Liao, W.-Q. Wang, B. Huang, and J. Xu, "Learning laplacian matrix for smooth signals on graph," in *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*. IEEE, 2019, pp. 1–5.

[45] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Laplacian matrix learning for smooth graph signal representation," in *2015 IEEE international conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 3736–3740.

[46] V. Kalofolias, "How to learn a graph from smooth signals," in *Artificial Intelligence and Statistics*. PMLR, 2016, pp. 920–929.

[47] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *Applied and Computational Harmonic Analysis*, vol. 44, no. 2, pp. 446–475, 2018.

[48] Y. H. Kim, "Qr factorization-based sampling set selection for bandlimited graph signals," *Signal Processing*, p. 107847, 2020.

[49] Y. Bai, F. Wang, G. Cheung, Y. Nakatsukasa, and W. Gao, "Fast graph sampling set selection using gershgorin disc alignment," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2419–2434, 2020.

[50] M. Coutino, S. P. Chepuri, and G. Leus, "Subset selection for kernel-based signal reconstruction," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4014–4018.

[51] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, "Eigendecomposition-free sampling set selection for graph signals," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2679–2692, 2019.

[52] G. Ortiz-Jiménez, M. Coutino, S. P. Chepuri, and G. Leus, "Sampling and reconstruction of signals on product graphs," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 713–717.

[53] Z. Wei, B. Li, and W. Guo, "Optimal sampling for dynamic complex networks with graph-bandlimited initialization," *arXiv preprint arXiv:1901.11405*, 2019.

[54] A. Chiumento, N. Marchetti, and I. Macaluso, "Energy efficient wsn: a cross-layer graph signal processing solution to information redundancy," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2019, pp. 645–650.

[55] J. Holm, F. Chiariotti, M. Nielsen, and P. Popovski, "Lifetime maximization of an internet of things (iot) network based on graph signal processing," *IEEE Communications Letters*, 2021.

[56] B. Gedik, L. Liu, and S. Y. Philip, "Asap: An adaptive sampling approach to data collection in sensor networks," *IEEE Transactions on Parallel and distributed systems*, vol. 18, no. 12, pp. 1766–1783, 2007.

[57] C. Liu, K. Wu, and J. Pei, "An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation," *IEEE transactions on parallel and distributed systems*, vol. 18, no. 7, pp. 1010–1023, 2007.

[58] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE transactions on signal processing*, vol. 63, no. 24, pp. 6510–6523, 2015.

[59] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Transactions on Signal Processing*, vol. 64, no. 18, pp. 4845–4860, 2016.

[60] R. P. Gupta, "Bounds on the chromatic and achromatic numbers of complimentary graphs," North Carolina State University. Dept. of Statistics, Tech. Rep., 1968.

[61] R. Wilson and J. J. Watkins, *Combinatorics: ancient & modern.* OUP Oxford, 2013.

[62] M. Aigner, "A characterization of the bell numbers," *Discrete mathematics*, vol. 205, no. 1-3, pp. 207–210, 1999.

[63] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging filed of signal processing on graphs," *IEEE Signal Processing Magazine*, 2013.

[64] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[65] N. Sulaimanov and H. Koeppl, "Graph reconstruction using covariance-based methods," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2016, no. 1, p. 19, 2016.

[66] F.-P. Chan, A.-C. Fu, and C. Yu, "Haar wavelets for efficient similarity search of time-series: with and without time warping," *IEEE Transactions on knowledge and data engineering*, vol. 15, no. 3, pp. 686–705, 2003.

[67] Y.-L. Wu, D. Agrawal, and A. El Abbadi, "A comparison of dft and dwt based similarity search in time-series databases," in *Proceedings of the ninth international conference on Information and knowledge management*, 2000, pp. 488–495.

[68] J. A. R. Macias and A. G. Exposito, "Efficient computation of the running discrete haar transform," *IEEE transactions on power delivery*, vol. 21, no. 1, pp. 504–505, 2005.

[69] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[70] P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers-a tutorial," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

[71] R. Pena, "Graph learning," https://github.com/rodrigo-pena/graph-learning.

[72] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[73] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 631–640.

[74] I. X. Leung, P. Hui, P. Lio, and J. Crowcroft, "Towards real-time community detection in large networks," *Physical Review E*, vol. 79, no. 6, p. 066107, 2009.

[75] K. Das, S. Samanta, and M. Pal, "Study on centrality measures in social networks: a survey," *Social network analysis and mining*, vol. 8, no. 1, pp. 1–11, 2018.

[76] B. Ruhnau, "Eigenvector-centrality—a node-centrality?" *Social networks*, vol. 22, no. 4, pp. 357–365, 2000.

[77] V. A. Traag, "Faster unfolding of communities: Speeding up the louvain algorithm," *Physical Review E*, vol. 92, p. 032801, 2015.

[78] J. Carbonell and J. Goldstein, "The use of mmr, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998, pp. 335–336.

[79] A. Sharma, K. K. Paliwal, S. Imoto, and S. Miyano, "Principal component analysis using qr decomposition," *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 6, pp. 679–683, 2013.

[80] K.-B. Yu, "Recursive updating the eigenvalue decomposition of a covariance matrix," *IEEE Transactions on Signal Processing*, vol. 39, no. 5, pp. 1136–1145, 1991.

[81] L. A. Rossman *et al.*, "EPANET 2: users manual," 2000.

[82] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[83] X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE transactions on signal processing*, vol. 63, no. 9, pp. 2432–2444, 2015.

[84] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 5445–5449.

[85] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2017, pp. 2826–2830.

[86] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

Paper C

Freshness on Demand: Optimizing Age of Information for the Query Process

Josefine Holm, Anders E. Kalør, Federico Chiariotti, Beatriz Soret, Søren K. Jensen, Torben B. Pedersen, and Petar Popovski

The paper has been published in the
ICC 2021-IEEE International Conference on Communications pp. 1–6, 2021.

The layout has been revised.

Abstract

AoI has become an important concept in communications, as it allows system designers to measure the freshness of the information available to remote monitoring or control processes. However, its definition tacitly assumed that new information is used at any time, which is not always the case. Instead instants at which information is collected and used are dependent on a certain query process. We propose a model that accounts for the discrete time nature of many monitoring processes, by considering a pull-based communication model *in which the freshness of information is only important when the receiver generates a query. We then define the QAoI, a more general metric that fits the pull-based scenario, and show how its optimization can lead to very different choices from traditional push-based AoI optimization when using a Packet Erasure Channel (PEC).*

1 Introduction

Over the past few years, the concept of information freshness has received a significant attention in relation to cyber-physical systems that rely on communication of various updates in real time. This has led to the introduction of *AoI* [1] as a metric that reflects the freshness at the receiver with respect to the sender, and denotes the difference between the current time and the time when the most recently received update was generated at the sender.

The first works to deal with AoI considered simple queuing systems, deriving analytical formulas for information freshness [2]. Followup works addressed AoI in specific wireless scenarios with errors [3] and retransmissions [4], or basing their analysis on live experiments [5]. The addition of more sources in the queuing system leads to an interesting scheduling problem, which aims at finding the packet generation rate that minimizes the age for the whole system [6]. Optimizing the access method and senders' updating policies to minimize AoI in complex wireless communication systems has been proven to be an NP-hard problem, but heuristics can achieve near-optimal solutions [7] by having sources decide whether an update is valuable enough to be sent (i.e., whether it would significantly reduce the AoI) [8]. The average AoI has been derived in slotted [9] and unslotted ALOHA [10], as well as in scheduled access [11], and the performance of scheduling policies has been combined with these access methods in [12].

However, the tacit assumption behind AoI, regardless of the system for which it is computed, has been that the receiver is interested in having fresh information *at any time*. In other words, this assumption works with *push-based communication*, in which a hypothetical application residing at the receiver has a *permanent query* to the updates that arrive at the receiver. The motivation for this paper starts by questioning this underlying assumption and generalizes the idea of AoI by considering the timing of the query process. This makes the communication between the sensor and receiver pull-

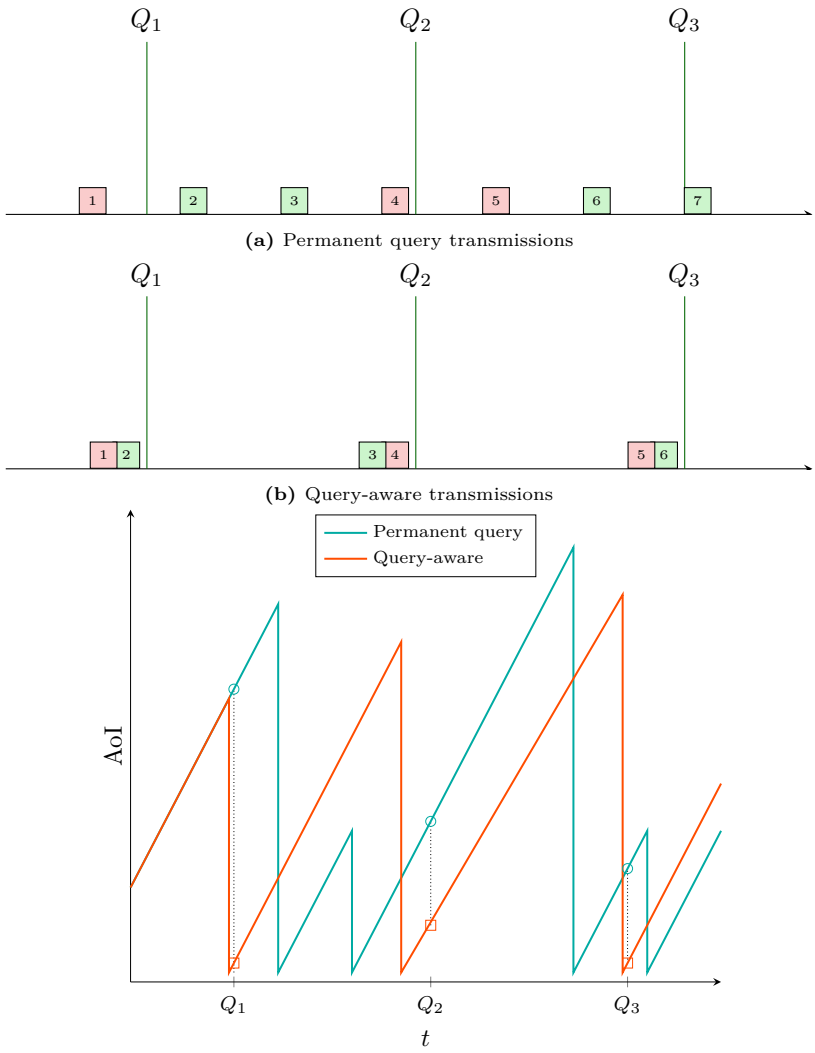
based, where the query can guide the communication strategy for the sensor updates.

The impact of the query-driven, pull-based communication model becomes immediately obvious with the (over)simplified example in Fig. C.1. The time is slotted and each packet, labeled $1, 2, \ldots 7$, takes one slot. Each update is generated immediately prior to the transmission. The queries $Q_1, Q_2, Q_3, \ldots$ arrive periodically, every 7-th slot. Furthermore, as an energy constraint, it is assumed that the sender can transmit on average one packet every 3 slots. Fig. C.1a shows the case in which the sender is oblivious to the query arrival process and distributes the transmissions evenly in time. Another strategy could be, in each slot, to decide to transmit with probability $1/3$ or stay silent otherwise; the important point is that this decision is made independently from the query process. Fig. C.1b shows the case in which communication is query-drive so the sender knows the query instants and optimizes the transmissions with respect to the timing of the query process, i.e., sends just before the query instants. In both cases the (red) packets $1, 4, 5$ are lost due to transmission errors. Fig. C.1c shows that the query-driven strategy is more likely to provide updates that are fresh when a query arrives, although its average AoI is worse at the instants in which there is no query.

Despite the deceptively simple insight offered by the example from Fig. C.1, the introduction of query-driven communication strategies does have a practical significance and introduces novel and interesting problems, as this paper shows. In fact, the assumption of a permanent query is relatively uncommon in the network control literature [13], which often uses periodic discrete time systems that poll the state of the monitored process at predefined intervals. Most network control systems are asynchronous, and use different sampling strategies that depend on the reliability of the connection and on the monitored process [14]. We define a *query arrival process* and consider the optimization of the communication process with respect to that arrival process. Furthermore, we define an QAoI metric which reflects the freshness in the instants when the receiver actually needs the data: having fresh data when the monitoring process is not asking for it does not provide any benefits to the system, as the information will not be used. Our model is also relevant for duty cycle-based applications, in which the sleeping pattern of the sensors are synchronized with the monitoring process.

This paper introduces models to analyze the difference in the communication strategies that should be used when the query arrival process is taken into account compared to the treatment of AoI in the context of a permanent query. In this initial work, we derive a Markov Decision Process (MDP) model for the problem with periodic queries and an erasure channel, and show that an optimization aimed at QAoI can significantly improve the perceived freshness with respect to classical models.

The remainder of the paper is organized as follows. We define the system model and the concept of QAoI in Sec. 2, and we formalize it as an MDP in Sec. 3. The setting and results of our simulations are described in Sec. 4, and Sec. 5 concludes the paper and presents some possible avenues of future work.



(a) Permanent query transmissions

(b) Query-aware transmissions

(c) Age for the two systems

Fig. C.1: Example of the difference between a system assuming a permanent query and one that is aware of the query arrival process. The same packets are lost (depicted in red) in both systems, and the markers indicate the age at the query arrival instants.

2 System model

We consider a scenario in which a wireless sensor generates updates at will and transmits them to an edge node over a wireless channel. The edge node receives queries from a server about the state of the sensor, e.g. as part of a monitoring or control process. The objective of this work is to maximize the freshness of the information used in the query responses while considering that the sensor is energy-constrained and needs to limit the number of transmissions to the edge node to prolong its lifetime.

2.1 Age of Information at Query

We consider a time-slotted system indexed by $t = 1, 2, \ldots$, and denote the time instances at which updates are successfully delivered to the edge node by $t_{u,1}, t_{u,2}, \ldots$. Following the common definition of AoI considered in the literature, e.g. [2, 6] we denote the AoI in time slot t by $\Delta(t)$, and define it as the difference between t and the time at which the last successfully received packet was generated:

$$\Delta(t) = t - \max_{i:t_{u,i} \leq t} t_{u,i}. \tag{C.1}$$

We will assume that $t_{u,1} = 0$ so that $\Delta(t)$ is well defined. An alternative, but equivalent definition can be obtained by introducing an indicator function $\psi(t)$, which is equal to 1 if a packet is successfully received in slot t and 0 otherwise:

$$\Delta(t) = \begin{cases} \Delta(t-1) + 1 & \text{if } \psi(t) = 0; \\ 1 & \text{if } \psi(t) = 1, \end{cases} \tag{C.2}$$

where $\Delta(0) = 0$.

Most work considers the problem of minimizing the long-term average of $\Delta(t)$. However, this is only one possibility in real monitoring and control systems: discrete-time systems involve queries in which the monitoring process samples the available information. To capture such applications, we introduce the QAoI metric, which generalizes AoI by sampling $\Delta(t)$ according to an arbitrary querying process, thereby considering only the instants at which a query arrives. We denote the query arrival times at the edge node by $t_{q,1}, t_{q,2}, \ldots$, and define the overall objective as minimizing the long-term expected QAoI defined as

$$\tau_\infty = \lim_{t \to \infty} \frac{1}{t} \mathbb{E}\left[\sum_{i:t_{q,i} \leq t} \Delta(t_{q,i}) \right]. \tag{C.3}$$

Although the query process may in general follow any random process, in this initial paper we limit the focus to the case in which the exact query instants are known in advance to the edge node and the sensor. This is for instance the case when the queries are periodic, or if the server repeatedly announces its next query instant.

2.2 Models for Communication and Query Arrivals

We assume that each update has a fixed size and is transmitted over a PEC with erasure probability ε. For simplicity's sake, in the following we refer to the success probability $p_s = 1 - \varepsilon$. Packets are instantaneously acknowledged by the receiver, so the sensor knows if a packet was erased or correctly received.

To model the energy-constrained nature of the node, we use a *leaky bucket* model, as commonly done in the literature [15]: we consider a bucket of tokens, which is replenished by a process which can generate tokens independently at each step with probability μ_b. The node can only transmit a packet if there are tokens in the bucket, and each transmission consumes one token. This model can fit an energy gathering node, as well as a general power consumption constraint on a battery-powered node, which should limit its number of transmissions in order to prolong its lifetime.

In this work, we assume the simplest possible query arrival process, with periodic queries every T_q steps. We assume that the sensor and receiver are synchronized, i.e., the sensor knows when the next query will come. While simple, this assumption is often realistic, as discrete time monitoring processes are often designed with a constant time step.

The model can be easily extended to more complex query arrival processes, and the process statistics can even be learned implicitly as part of the optimal strategy, as long as it is consistent. If we follow the definitions from Sec. 2.1, the strategies to minimize AoI and QAoI coincide in the memoryless case in which the query arrival process is Poisson or when the query arrival process is much faster than the sensor, i.e., when there is a query in each time slot.

3 MDP formulation and problem solution

In the following, we will model the two communication scenarios described in the next paragraph as MDPs, which we will then proceed to solve. An MDP is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a set of transition probabilities $p_a(s, s') = P(s_{t+1} = s' | a_t = a, s_t = s)$, and an instantaneous reward function $r(s, a, s')$, which represents the immediate reward when taking action a and transitioning from state s to state s'. The model can be used to represent two different systems: a Permanent Query (PQ) system, which minimizes the traditional AoI, and a Query Arrival Process Aware (QAPA) system, which minimizes the QAoI, only caring about the instants when a query arrives. These two systems can use the same state and action spaces, and only differ in the reward function that they use.

Decisions are made at every slot, as the sensor can either keep silent or send a packet. Consequently, the action space is $\mathcal{A} = \{0, 1\}$. As the aim of the QAPA agent is to minimize the QAoI, the state should include the current age $\Delta(t)$, as well as the number of slots $\sigma(t)$ until the next query. Additionally, the agent should know the

number of available tokens, $b(t)$, as it will influence its decision whether to transmit. If the number of tokens is 0, the sensor is blocked from transmitting until a token is generated. The state space can then be defined as $\mathcal{S} = \mathbb{N} \times \{0, \dots, T_q - 1\} \times \mathbb{N}$, where $\mathbb{N}$ indicates the set of strictly positive integers. A state s_t is given by the tuple $(\Delta(t), \sigma(t), b(t))$. Each element in the state tuple evolves independently between time steps, so in the following we describe the state dynamics one by one.

The AoI increases by one between each slot unless the node decides to transmit and the packet is successfully received, with probability p_s, in which case the AoI is reduced to one in the subsequent slot. The non-zero transition probabilities are thus described by

$$\Pr(\Delta(t+1) = \delta | \Delta(t), a_t) = \begin{cases} a_t p_s & \text{if } \delta = 1; \\ 1 - a_t p_s & \text{if } \delta = \Delta(t) + 1; \\ 0 & \text{otherwise,} \end{cases} \tag{C.4}$$

where a_t is the action at time t, which equals zero if the sensor is silent and one if it transmits. The time until the next query $\sigma(t)$, is deterministic and independent of the action, and decreases by one until it reaches zero, at which point it is reset to T_q. Assuming that the first query happens at time $t = T_q$, the value of $\sigma(t)$ can be written

$$\sigma(t) = T_q - t \pmod{T_q}. \tag{C.5}$$

Finally, the number of tokens in the next slot depends on whether a new token is generated and whether the sensor transmits, in which case, it uses one token. The transition probability from $b(t)$ to $b(t+1)$ is:

$$p(b(t+1) = b + i | b(t), a_t) = \begin{cases} \mu_b & \text{if } i = 1 - a_t; \\ 1 - \mu_b & \text{if } i = -a_t; \\ 0 & \text{otherwise.} \end{cases} \tag{C.6}$$

We define two cost functions; one for the PQ system, which does not depend on the query instant and will be used as baseline, and one for the QAPA system, in which the cost is only considered when a query arrives. In the baseline PQ model, the cost is given by the AoI in any slot:

$$c_{\text{PQ}}(s_t, a_t, s_{t+1}) = \Delta(t+1). \tag{C.7}$$

However, in the QAPA system, the cost is the AoI when a query arrives:

$$c_{\text{QAPA}}(s_t, a_t, s_{t+1}) = \begin{cases} \Delta(t+1) & \text{if } \sigma(t+1) = 0; \\ 0 & \text{otherwise.} \end{cases} \tag{C.8}$$

In both cases, the objective is to find a policy π^* that minimizes the long-term cost. In this initial work, we limit ourselves to consider the discounted case, which benefits from

strong convergence guarantees, and defer the case with undiscounted costs to future work. Specifically, we solve:

$$\pi^* = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(t)|\pi\right], \tag{C.9}$$

where $\lambda < 1$ is the discount factor.

We can now proceed to solve the MDP for the two systems we have defined using policy iteration, as described in [16, Ch. 4]. In order to apply the algorithm, we need to truncate the problem to a finite MDP. We do so by defining a maximum age $\Delta_{\max}$ and a token bucket size B: once the age or the number of tokens in the bucket reach the maximum, they cannot increase further. As long as the maximum values are sufficiently large, they are not reached during normal operation and this simplification does not affect the optimal policies or their performance.

The policy iteration algorithm has two steps: 1) policy evaluation and 2) policy improvement which are repeated until convergence. To solve the proposed problem we initialize the policy with zeros i.e. the policy where we never send any updates, and the value to be larger than we expect from a reasonable policy.

1. The policy is evaluated using

$$v_\pi(s) = \sum_{s'} p(s', c|s, a)\left(c + \lambda v_\pi(s')\right). \tag{C.10}$$

   for all s, where s is the current state, s' is the new state, a is the action, and c is the cost from either (C.7) or (C.8).

2. The policy is improved by evaluating

$$q_\pi(s, a) = \sum_{s'} p(s', c|s, a)\left(c + \lambda v_\pi(s')\right) \tag{C.11}$$

   for all a. If $q_\pi(s, a) > v_\pi(s)$ we substitute a into the policy. This is repeated for all s.

Policy iteration is guaranteed to converge to the optimal policy [17] in finite-state MDPs with finite reward. As mentioned above, we truncated the age and token bucket size to make the MDP finite, so the conditions to use the algorithm apply.

4 Numerical results

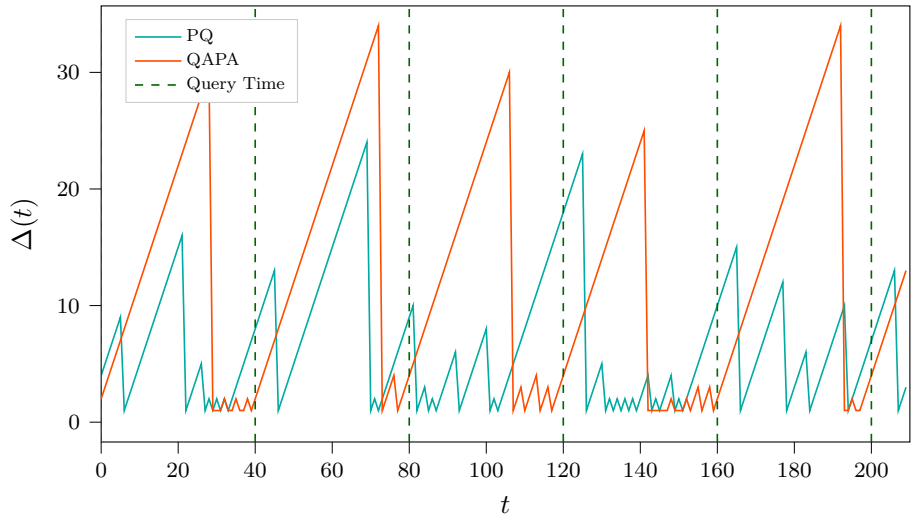
This section presents Monte Carlo evaluations of the policies obtained using the MDP described in Section 3. Although, the methods in Section 3 can be applied to any query

process, throughout the evaluation we will consider queries that occur periodically, at a fixed time interval T_q. Furthermore, we truncate the MDP at a maximum age of $\Delta_{\max} = 100 \times T_q$ and a maximum token bucket size of $B = 10$, and we use a discount factor $\lambda = 0.75$. We use the term AoI to refer to the age at any time and QAoI for the age sampled at the query instants.

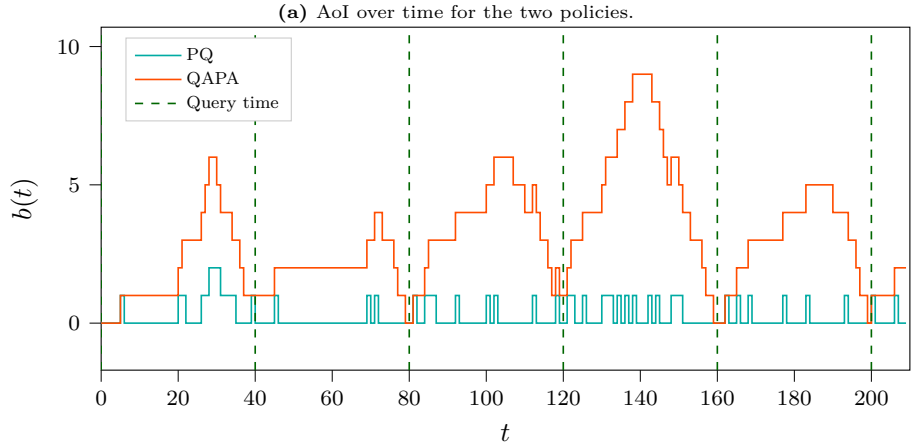
We start by exploring the temporal dynamics of the AoI process obtained using the PQ and the QAPA policies. Recall that PQ is optimized to achieve a low AoI independent of the query process, while QAPA minimizes the AoI at the query times, using cost functions (C.7) and (C.8), respectively. Fig. C.2a shows the AoI for queries occurring periodically every $T_q = 40$-time slots as indicated by the vertical lines, a packet error probability of $\varepsilon = 0.2$, and a token rate $\mu_b = 0.2$. It is seen that the PQ policy reduces the AoI approximately uniformly across time, while the QAPA policy consistently tries to reduce the AoI in the slots immediately prior to a query, so that the AoI is minimized when the query arrives. This is reflected in Fig. C.2b, which shows that the QAPA policy accumulates energy when the next query is far in the future, unlike PQ. A consequence of this is that the QAPA policy generally has a slightly higher average AoI than the PQ policy, but the QAoI of the QAPA is significantly lower than that of the PQ policy.

The initial observations from Fig. C.2 can be confirmed by the distribution of the AoI as a function of the time since the last query, as illustrated in Figs. C.3 and C.4. Figs. C.3a and C.4a show the probability mass function of the AoI conditioned on various time instants $t \mod T_q$, while Figs. C.3b and C.4b show the Cumulative Distribution Function (CDF) of the overall AoI and QAoI. In the scenario with low error probability, $\varepsilon = 0.2$, the AoI distribution of the PQ policy is uniform across time (upper plot in Fig. C.3a), while the QAPA policy has an increasing age as time since the query passes, but a far lower age right before and at the query instant, $t \mod T_q = 0$ (lower plot in Fig. C.3a). The resulting CDF in Fig. C.3b reveals, as expected, that the AoI and the QAoI are equivalent for the PQ policy, as the distribution is the same at any time instant. However, for the QAPA policy, the QAoI is significantly lower than the AoI, while the AoI is often larger than the PQ policies. This is because the QAoI is only measured at the query instants, at which the age of the QAPA policy is minimized. Due to the energy constraint, this comes at the cost of a generally higher age, causing a higher AoI measured at each time instant. Finally, the staircase appearance in the CDF is because the queries happen periodically. If the queries came at variable (but known in advance) intervals, the CDF would be smoother, maintaining QAPA's performance advantage.

The same observations apply for the scenario with high error probability, $\varepsilon = 0.7$, shown in Figs. C.4a and C.4b. Although the AoI and QAoI are higher due to the high packet error rate, the applied policies are similar. The gain that the QAPA policy achieves by clustering its transmissions close to the query instant is clearly reflected in Fig. C.4a where, although there is a significant probability that the packet immediately

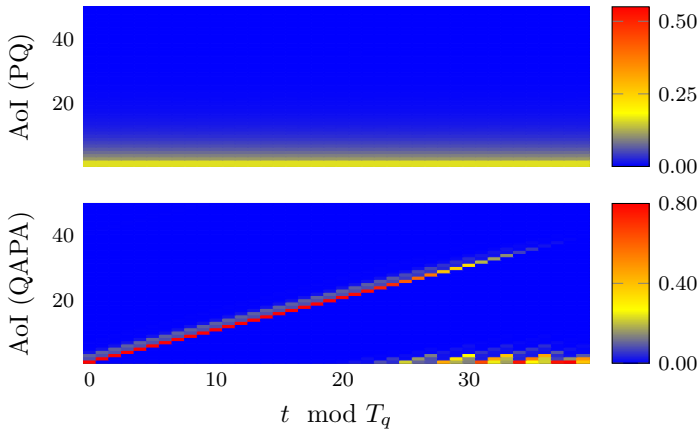


(a) AoI over time for the two policies.

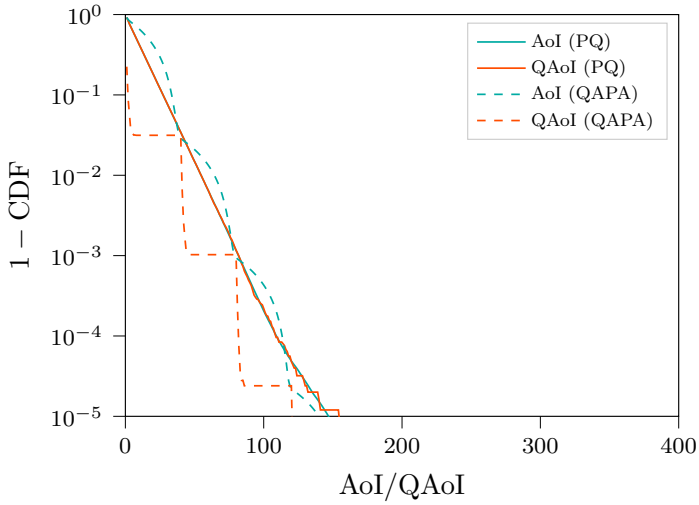


(b) Available tokens over time for the two policies.

Fig. C.2: AoI dynamics of the PQ and QAPA policies for $T_q = 40, \mu_b = 0.2, \varepsilon = 0.2$. The PQ policy generally has a lower AoI, but the QAPA policy minimizes the AoI at the query instants.

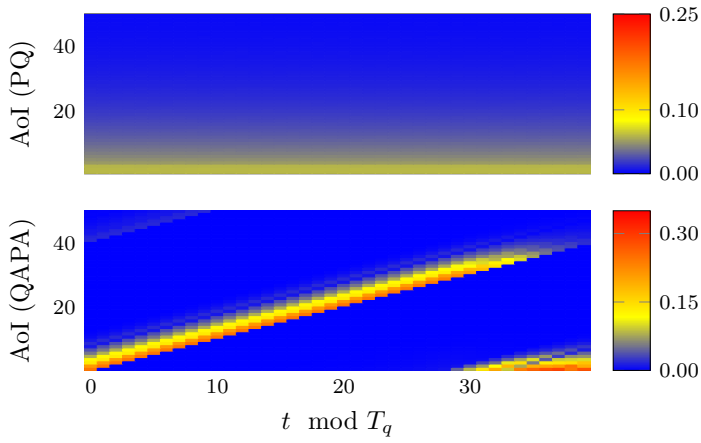


(a) PQ (upper) and QAPA (lower) density, $\varepsilon = 0.2$.

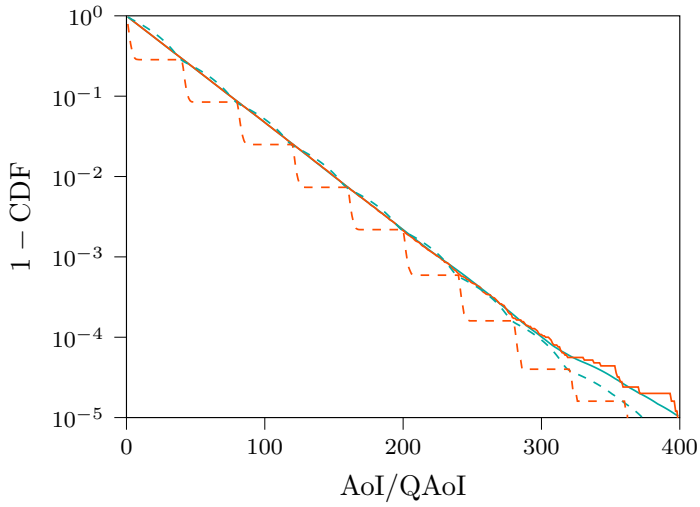


(b) Complementary CDF, $\varepsilon = 0.2$.

Fig. C.3: AoI distributions and CDFs for PQ and QAPA for $T_q = 40, \mu_b = 0.1$ and $\varepsilon = 0.2$. (a): AoI distribution for the PQ and QAPA at various time instances $t \mod T_q$ for $\varepsilon = 0.2$. PQ achieves low AoI at all times, QAPA ensures that the AoI is low at the query instants, i.e. $t \mod T_q = 0$. (b): Complementary CDF of the AoI and QAoI achieved by the two policies for $\varepsilon = 0.2$. Generally, the QAPA policy has lower QAoI but higher AoI than the PQ policy.



(a) PQ (upper) and QAPA (lower) density, $\varepsilon = 0.7$.



(b) Complementary CDF, $\varepsilon = 0.7$.

Fig. C.4: AoI distributions and CDFs for PQ and QAPA for $T_q = 40, \mu_b = 0.1$ and $\varepsilon = 0.7$. (a): AoI distribution for the PQ and QAPA at various time instances $t \mod T_q$ for $\varepsilon = 0.7$. PQ achieves low AoI at all times, QAPA ensures that the AoI is low at the query instants, i.e. $t \mod T_q = 0$. (b): Complementary CDF of the AoI and QAoI achieved by the two policies for $\varepsilon = 0.7$. Generally, the QAPA policy has lower QAoI but higher AoI than the PQ policy.

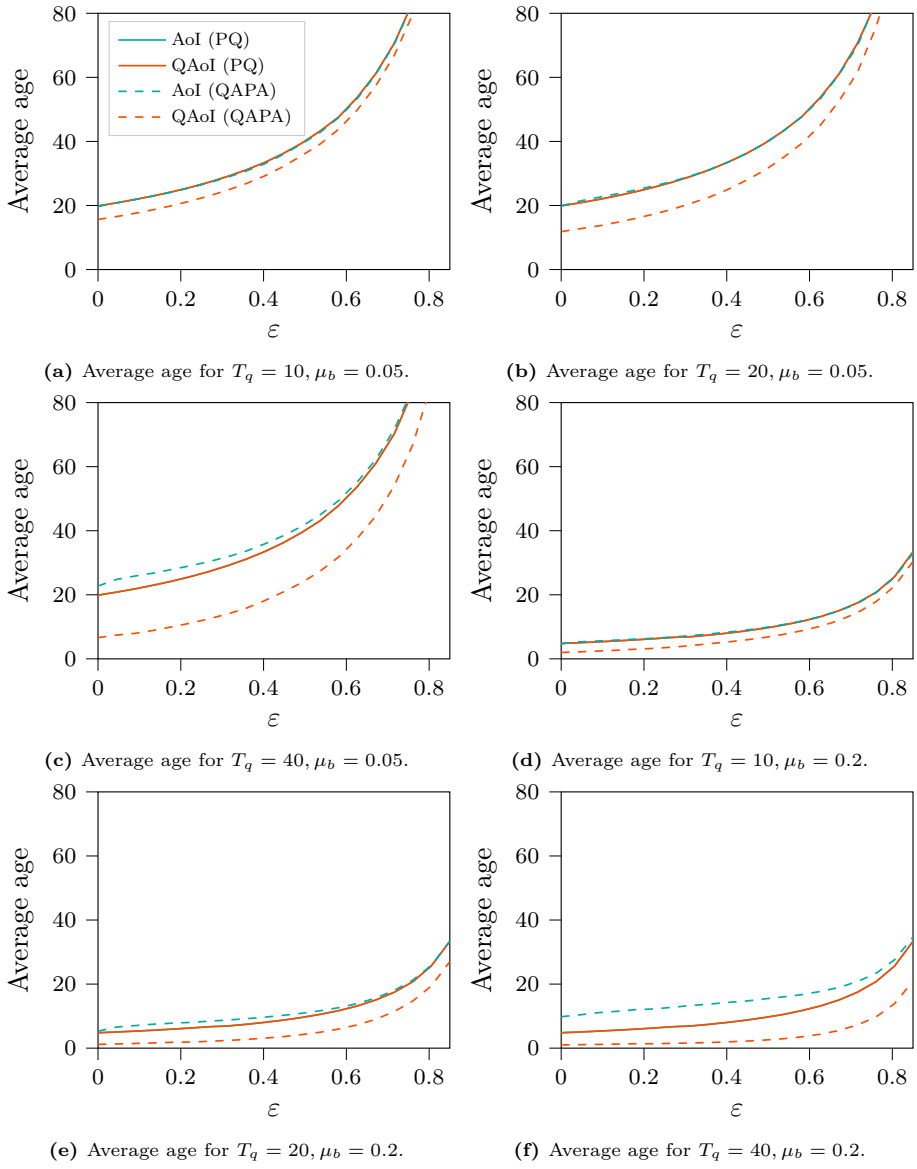
prior to the query is lost, the AoI distribution at $t \mod T_q = 0$ is still concentrated close to one.

We close the section by studying how the average AoI and QAoI changes with the packet error probability ε for various choices of the parameters, shown in Fig. C.5. For all cases, the QAPA policy achieves the lowest QAoI, while the PQ policy achieves the lowest AoI. When the query period, T_q, is low, the difference between AoI and QAoI is relatively small, as is the difference between the two policies. Intuitively, this is because the query instants, which are prioritized by the QAPA policy, are more frequent, making the two problems more similar. If we set $T_q = 1$, the two policies would coincide. As a result, awareness of the query arrival process becomes more important when queries are rare, i.e., when T_q is large: this is clear from the large gap between the average QAoI achieved by QAPA and by PQ in Fig. C.5c and Fig. C.5f. The upper row, Fig. C.5a-C.5c, shows the results for $\mu_b = 0.05$, i.e., when a new token is generated on average every 20 time slots. When $T_q = 10$ (C.5a), the token period becomes a limiting factor, and both the AoI and QAoI are relatively high even for low values of ε. In particular, in the error-free case when $\varepsilon = 0$, the average QAoI cannot be lower than $(1 + 11)/2 = 6$, which is achieved by transmitting an update prior to every second query. Interestingly, the impact of the energy limit becomes less significant for the QAPA policy as the time between queries increases: by saving up tokens until right before the query, this policy can significantly reduce the QAoI, at the cost of a higher AoI. On the other hand, the PQ policy does not benefit from this increase, as it is oblivious of the query arrival frequency. When tokens are generated faster, at rate $\mu_b = 0.2$, as shown in Fig. C.5d-C.5f, the AoI and the QAoI are generally lower, since more frequent transmissions are allowed.

5 Conclusions and future work

In this paper, we proposed a new metric for information freshness, which we dubbed QAoI: unlike standard AoI for push-based communication, this metric can be used for pull-based communication in which the monitoring process is not always listening, but sends queries when it is interested in the information. With the proposed model and subsequent MDP solution, we show the benefit of optimizing the transmission policy using the available knowledge on the query arrival process. Our results show that the standard PQ optimization, which minimizes AoI at any instant, can be very different from a QAPA policy that optimizes QAoI by concentrating its transmissions right before it expects a new query.

We are considering several avenues of future work. (i) A formal derivation of the QAoI in simple queuing systems. (ii) Modeling of more complex query processes with stochastic timing, which would require the sensors to learn the nature of the query arrival process online.



(a) Average age for $T_q = 10, \mu_b = 0.05$.

(b) Average age for $T_q = 20, \mu_b = 0.05$.

(c) Average age for $T_q = 40, \mu_b = 0.05$.

(d) Average age for $T_q = 10, \mu_b = 0.2$.

(e) Average age for $T_q = 20, \mu_b = 0.2$.

(f) Average age for $T_q = 40, \mu_b = 0.2$.

Fig. C.5: Average AoI and QAoI for the two systems for different values of T_q and μ_b.

Acknowledgment

This work has, in part, been supported by the Danish Council for Independent Research (Grant No. 8022-00284B SEMIOTIC).

References

[1] 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.

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

[3] K. Chen and L. Huang, "Age-of-information in the presence of error," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jul. 2016, pp. 2579–2583.

[4] R. Devassy, G. Durisi, G. C. Ferrante, O. Simeone, and E. Uysal, "Reliable transmission of short packets through queues and noisy channels under latency and peak-age violation guarantees," *IEEE Journ. on Selected Areas in Communications*, vol. 37, no. 4, pp. 721–734, Feb. 2019.

[5] H. B. Beytur, S. Baghaee, and E. Uysal, "Measuring age of information on real-life connections," in *27th Signal Processing and Communications Applications Conf. (SIU)*. IEEE, Apr. 2019.

[6] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," *IEEE Trans. on Mobile Computing*, Dec. 2019.

[7] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. on Information Theory*, vol. 63, no. 11, pp. 7492–7508, Nov. 2017.

[8] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2015, pp. 3008–3012.

[9] R. D. Yates and S. K. Kaul, "Status updates over unreliable multiaccess channels," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2017, pp. 331–335.

[10] R. D. Yates and S. K. Kaul, "Age of information in uncoordinated unslotted updating," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2020, pp. 1759–1764.

[11] R. Talak, S. Karaman, and E. Modiano, "Distributed scheduling algorithms for optimizing information freshness in wireless networks," in *19th Int. Worksh. on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, Jun. 2018.

[12] X. Chen, K. Gatsis, H. Hassani, and S. S. Bidokhti, "Age of information in random access channels," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2020, pp. 1770–1775.

[13] L. Zhang and D. Hristu-Varsakelis, "Communication and control co-design for networked control systems," *Automatica*, vol. 42, no. 6, pp. 953–958, Jun. 2006.

[14] D. Zhang, P. Shi, Q.-G. Wang, and L. Yu, "Analysis and synthesis of networked control systems: A survey of recent advances and challenges," *ISA transactions*, vol. 66, pp. 376–392, Jan. 2017.

[15] V. Raghunathan, S. Ganeriwal, M. Srivastava, and C. Schurgers, "Energy efficient wireless packet scheduling and fair queuing," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 3, no. 1, pp. 3–23, Feb. 2004.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[17] R. A. Howard, *Dynamic programming and Markov processes*. John Wiley, 1960.

Paper D

Query Age of Information: Freshness in Pull-Based Communication

Federico Chiariotti, Josefine Holm, Anders E. Kalør, Beatriz Soret, Søren K. Jensen, Torben B. Pedersen, and Petar Popovski

The paper has been published in the
IEEE Transactions on Communications Vol. 70, Nr. 3, pp. 1606–1622, 2022.

The layout has been revised.

Abstract

AoI has become an important concept in communications, as it allows system designers to measure the freshness of the information available to remote monitoring or control processes. However, its definition tacitly assumes that new information is used at any time, which is not always the case: the instants at which information is collected and used may be dependent on a certain query process, and resource-constrained environments such as most IoT use cases require precise timing to fully exploit the limited available transmissions. In this work, we consider a pull-based communication model *in which the freshness of information is only important when the receiver generates a query: if the monitoring process is not using the value, the age of the last update is irrelevant. We optimize the QAoI, a metric that samples the AoI at relevant instants, better fitting the pull-based resource-constrained scenario, and show how this can lead to very different choices. Our results show that QAoI-aware optimization can significantly reduce the average and worst-case perceived age for both periodic and stochastic queries.*

1 Introduction

Over the past few years, the concept of information freshness has received a significant attention in relation to cyber-physical systems that rely on communication of various updates in real time. This has led to the introduction of *AoI* [1] as a measure that reflects the freshness at the receiver, and denotes the difference between the current time and the time when the most recently received update was generated at the sender.

In a common model for AoI-sensitive systems, a wireless-equipped sensor measures a physical process and transmits its readings using a wireless link to a destination, where a monitor processes the received information. On the other hand, we study the effect of the existence of a *query process* at the receiver in a resource-limited scenario in which a single sensor is constrained in how often it can transmit, either due to energy considerations or duty cycle limitations. In most works in the literature, the tacit assumption behind AoI has been that the monitor at the receiver is interested in having fresh information at any time. In other words, the model assumes a *push-based communication*, in which a hypothetical application residing at the monitor has a *permanent query* to the updates that arrive at the receiver. Our work considers a *pull-based communication* model, in which the query arrival process can guide the communication strategy by, for example, reading the sensor and transmitting the updates immediately before the query times. To design for pull-based communications, we introduce an age metric named QAoI, which represents the age of the information available to the receiver in the instant when it actually needs it. This new metric is similar to the Effective Age of Information (EAoI), presented in [2], which studied the effect of queries in a different scenario, with multi-

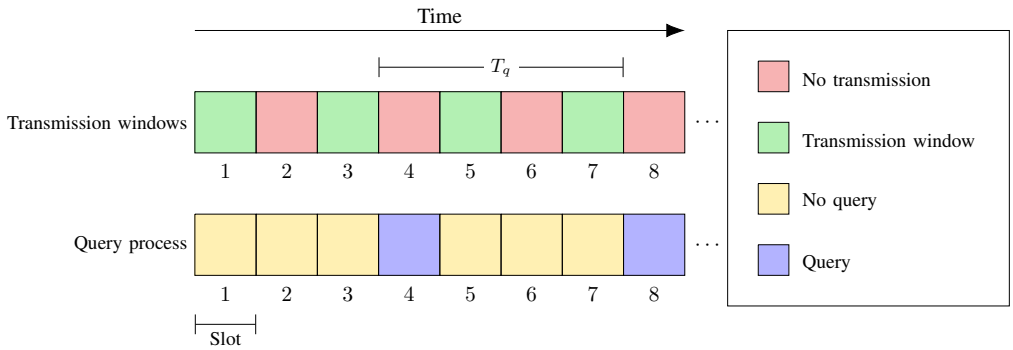


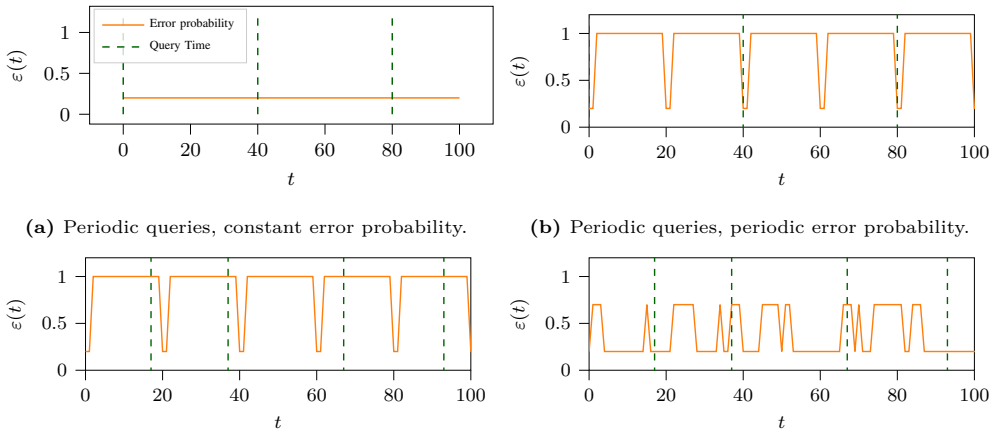
Fig. D.1: Diagram of a satellite IoT-based pull-based communication model.

user scheduling and a constant channel. The QAoI metric and the system optimization based on it is tested in a scenario with *limited link availability* at the source, as our system model considers both when it is possible to transmit and when it is convenient to do so.

One can think of several scenarios that would fit this pull-based model, as most monitoring and control applications operate over discrete time intervals, or only activate to react to some external trigger or user input. An interesting practical use case for this concept is represented by the Satellite IoT, which connects sensors in remote areas to the wider Internet through the use of Geosynchronous Earth Orbit (GEO) or Low Earth Orbit (LEO) satellites, and is still a mostly unexplored setting for the AoI literature [3, 4]. Fig. D.1 shows an example of the transmission and query timing with periodic intervals: the transmitter can only send a packet in the green slots, and the application queries the received result in the blue slots. In the push-based communication model, the transmitter should optimize for freshness at any time, while in the pull-based model, if there is a successful transmission in, e.g., slot 7, any transmission in slot 5 will then be useless to the application, as it will never see it. The transmitter will try to send packets as close to the query instant as possible, even if this results in a larger age in between queries.

In our model, the channel between the sensor and the intermediate cache, i.e., the satellite uplink and downlink in the Satellite IoT scenario, is abstracted as a PEC, whose erasure probability can be either constant or time-varying. The query arrival process is, in general, a stochastic process. In our simulations, we consider four scenarios, which can correspond to three different Satellite IoT use cases:

- *Periodic queries and constant channel*: this is the simplest case, in which queries are deterministic and periodic and the channel error probability is constant over time. This model can represent a GEO monitoring application, in which the satellite is always in the same position relative to the ground and the remote monitoring application simply logs the data by querying the ground station at



(a) Periodic queries, constant error probability.

(b) Periodic queries, periodic error probability.

(c) Stochastic queries, periodic error probability.

(d) Stochastic queries, stochastic error probability.

Fig. D.2: Scenarios considered in the simulations.

predetermined intervals. This corresponds to the scenario in Fig. D.2a, in which $\varepsilon(t)$ represents the channel error probability at time t.

- *Periodic queries with a predictable channel*: in this case, we introduce some complexity in the channel behavior by having a time-varying error probability, while maintaining a deterministic and periodic query arrival process. This scenario can represent a LEO remote monitoring application, in which the sensor is not served by a GEO satellite, but by a constellation of LEO satellites, whose orbits bring them outside the coverage range of the sensor: when there are no visible satellites, the packet error probability is 1. The sensors must then transmit its data when at least one satellite is passing over it. The orbits of the satellites can be computed in advance, so these periods of availability are known, but the sensor will be constrained in its scheduling decisions. This corresponds to the scenario in Fig. D.2b. Another possible example of this model would be a wireless scenario with recurring outages, during which the error probability for updates sharply increases.

- *Stochastic queries with a predictable channel*: in this case, the channel error probability can change over time and queries arrive according to a stochastic process, so the sensor will need to optimize the expected QAoI, considering the probability of queries arriving in the near future. This scenario can represent a human-in-the-loop LEO monitoring application, in which queries are driven by the behavior of the user, and are then only partially predictable. This corresponds to the scenario in Fig. D.2c.

- *Stochastic queries with a stochastic channel*: this is the most general case, in which

queries are stochastic, and the channel error probability is not predictable, but follows a stochastic process such as a Markov chain. This case can represent a general wireless channel, which does not have pre-computed satellite passes, but depends on the propagation conditions, and is shown in Fig. D.2d. It corresponds to a general wireless edge system getting queries from the Cloud.

These four examples are described in more detail and adapted to the communication model in Sec. 6. Besides the satellite IoT application of Fig. D.1, our model can fit several other monitoring applications, and the formulation is fully general for relaying scenarios with an intermediate cache node fulfilling requests from the end user. A generic example includes queries that are periodically/regularly sent from a central cloud to an edge node.

In this work, we model a scheduling problem for a resource-constrained sensor as a MDP, showing the difference between a query-aware scheduler and a legacy one that tries to minimize AoI at any instant: in the most general case, we consider the query arrival process and the channel state to be driven by two independent Markov chains. The model in this paper extends the framework we presented in [5], which only considered a simple scenario with a constant channel and periodic queries. While previous works on AoI often dealt with limitations on the link availability, because of congestion, energy, or propagation constraints, but to the best of our knowledge, this is the first work to combine limitations on the channel availability with AoI in pull-based communication.

Our simulation results show that awareness of the query process can give significant gains in terms of both the average QAoI and its higher percentiles, which represent a worst-case result and are critical for reliability-oriented applications, improving the performance of monitoring systems even in the most general case. The query-aware optimization often incurs a cost in terms of AoI, as the scheduling strategy that optimizes freshness when a query arrives will often let the age increase when the probability of a query arriving is low,

The rest of this paper is divided as follows: first, Sec. 2 presents the state of the art on scheduling and AoI minimization. We then present the QAoI metric and our communication system model in Sec. 3, formulating it as an MDP and finding the optimal policies in Sec. 5. We then present our simulation and its results in Sec. 6, considering a simple system first and gradually increasing its complexity. Finally, we conclude the paper and describe some possible avenues of future work in Sec. 7.

2 Related work

Over the last few years, AoI [6] has attracted a significant amount of interest from the research community, as it represents a more relevant metric than latency for the ongoing monitoring and control of processes over a network. Most works in the literature deal with AoI in queuing systems, examining different scheduling policies. Some recent works

have proven that preemption, i.e., removing packets already in the queue in favor of newer ones with more up to date information, can give significant advantages in terms of average age [7], even when multiple $M/M/1$ queuing systems in tandem are involved [8]. However, preemption can be suboptimal for different service time distributions, as the decision over whether to preempt or to continue the ongoing transmission becomes more complex [9].

Other works addressed AoI in specific wireless scenarios with errors [10] and retransmissions [11], or basing their analysis on live experiments [12]. The addition of more sources in the queuing system leads to an interesting scheduling problem, which aims at finding the packet generation rate that minimizes the age for the whole system [13]. Optimizing the access method and senders' updating policies to minimize AoI in complex wireless communication system has been proven to be an NP-hard problem, but heuristics can achieve near-optimal solutions [14] by having sources decide whether an update is valuable enough to be sent (i.e., whether it would significantly reduce the AoI) [15]. The average AoI has been derived in slotted [16] and unslotted ALOHA [17], as well as in scheduled access [18], and the performance of scheduling policies has been combined with these access methods in [19].

The scheduling problem can be formulated both for multiple sources, in which case the scheduling problem involves balancing the ages of the different sources while avoiding interference [20], or for a single source with resource constraints: usually, these constraints are in the form of limited energy availability or enforced duty cycles. Energy harvesting nodes are considered in [21], which derives a near-optimal threshold-based heuristic that can work without knowledge of future energy generation, and by [22], which derives the optimal policy for nodes with infinite as well as finite batteries. The trade-off between energy and freshness is examined explicitly in [23], while [24] considers a noisy channel as well, in which updates can be randomly erased. A more complex scenario, which includes a Hybrid Automated Repeat Request (HARQ) channel as well as an energy harvesting node with a finite battery, is considered in [25], which models the problem as an MDP and finds the optimal policy with reinforcement learning. Another interesting case for the scheduling problem is AoI minimization in drone networks, in which drones have to move back and forth between the sensing location and the base station [26]: finding the correct balance to minimize AoI and energy expenditure is a non-trivial problem in this scenario.

To the best of our knowledge, most of the literature so far has adopted a push-based model, in which updates are always relevant to the monitoring process. We are aware of only a few other works that consider a pull-based model: the one most similar to this work [2] considers a server updating multiple users, using a metric called EAoI, which is similar to QAoI, although in their case the system is not constrained by energy considerations, but by the presence of multiple sensors that need to be scheduled appropriately. The effect of queries is also modeled slightly differently: in our case, there is a strict ordering between transmissions and queries, and the response to a query is

Symbol	Description
$t_{u,i}$	Delivery time of the i-th update
$\Delta(t)$	AoI at time t
Δ_∞	Long-term expected AoI
$t_{q,i}$	Time of the i-th query
$\mathcal{S}_q$	State space of the query process
$\mathcal{Q}$	Set of states in which a query arrives
P_q	Transition matrix of the query process
$\mathcal{S}_e$	State space of the error probability process
P_e	Transition matrix of the error probability process
$\varepsilon(s_e)$	Packet error probability for state s_e
T_e	Packet error probability period
T_q	Query arrival period
$\mathcal{S}$	State space of the scheduling MDP
$\mathcal{A}$	Action space of the scheduling MDP
$p_a(s, s')$	Probability to go from s to s' for action a
$r(s, a, s')$	Instantaneous reward
$b(t)$	Number of available tokens at time t
$c(s_t, a_t)$	Long-term expected cost
λ	Cost discount factor
π	Action policy
$v_\pi(s_t)$	Expected state value with policy π
ε_0	Error probability during the satellite pass

Table D.1: Notation definitions.

always sent immediately. On the other hand, EAoI considers the possibility of a *delayed response* if a transmission from the sensor is scheduled but not yet received. In our work, we focus on the optimization of the connection between the sensor and the intermediate cache, considering significant communication constraints and a more challenging IoT scenario. The difference between our metric and EAoI is shown in Sec. 4. Another work, first presented by Li *et al.* in [27] and later expanded in [28], considers age not to matter unless and until a request for the information is generated. However, Li *et al.*'s work does not consider the effects of scheduling in a regime with limited transmission opportunities, but rather tries to provide freshness to the user by combining multiple replications of the sensor value over multiple servers. The innovation from [28] can be combined with ours with limited adaptations to the two models, resulting in a joint optimization of the whole end-to-end scheduling. Another work also models requests in the optimization function [29], but it only deals with memoryless request processes, which (as we will describe in the introduction) lead to a solution that is equivalent to standard AoI minimization. The extended version of that paper [30] considers more complex scenarios with partial battery knowledge, but still uses the same memoryless request model. Finally, a recent work by Xu *et al.* [31] also considers a memoryless request process, but considers a mix between traditional AoI and query-aware metrics. This is a significant difference from QAoI and related metrics such as EAoI, and it leads

to a highly different optimization, which will prioritize users with a less active request process.

3 System model

We now define a simple system model and consider the QAoI metric. The notation in the following sections is summarized in Table D.1.

We consider a time-slotted system indexed by $t = 1, 2, \ldots$, and denote the time instants at which updates are successfully delivered to the edge node as $t_{u,1}, t_{u,2}, \ldots$. The source can be sampled at any time, and fresh information is always available, a condition known as zero-wait sampling. Following the common definition of AoI considered in the literature, e.g. [6, 13] we denote the AoI in time slot t by $\Delta(t)$, and define it as the difference between t and the time at which the last successfully received packet was generated:

$$\Delta(t) = t - \max_{i:t_{u,i} \leq t} t_{u,i}. \tag{D.1}$$

We will assume that $t_{u,1} = 0$ so that $\Delta(t)$ is well defined. An alternative, but equivalent definition can be obtained by defining the time-varying variable u_t that takes the value 1 if a new update is received at the edge node in time slot t, and 0 otherwise:

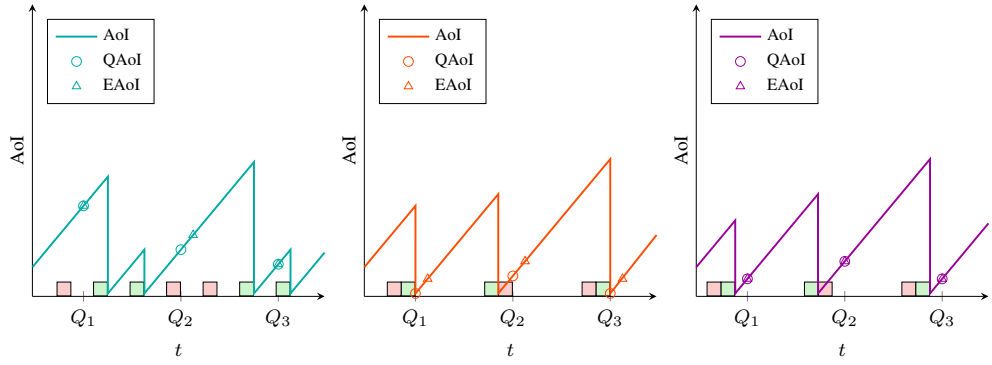
$$\Delta(t) = \begin{cases} \Delta(t-1) + 1 & \text{if } u_t = 0 \\ 1 & \text{if } u_t = 1 \end{cases} \tag{D.2}$$

where $\Delta(0) = 0$. This definition of AoI, as given in [6], considers the freshness of information at any given point in time. The long-term expected AoI Δ_∞ is given by:

$$\Delta_\infty = \limsup_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=1}^{T} \Delta(t)\right]. \tag{D.3}$$

This formulation does not include any weighting, assuming that all time steps are equally important. This is reasonable if the monitoring system is either continuous or much faster than the update generating process and communication system, i.e., can be considered as essentially continuous. However, this is only one possibility in real monitoring and control systems: discrete-time systems involve queries in which the monitoring process samples the available information. To capture such applications, we introduce the QAoI metric, which samples $\Delta(t)$ according to an arbitrary querying process, thereby considering only the instants at which a query arrives. In this case, we can consider long-term AoI as a special case of QAoI in which queries arrive at every time instant.

Naturally, in order for an update to be received successfully in slot t, we need to transmit it: the *policy* to transmit an update is a function $\pi : \mathcal{S} \to \{0, 1\}$, where $\mathcal{S}$ is a state space and an update is transmitted if the policy outputs 1.



(a) AoI optimization. **(b)** QAoI optimization. **(c)** EAoI optimization.

Fig. D.3: Example of the difference between a system assuming a permanent query and one that is aware of the query arrival process. The same packets are lost in all systems (correctly received packets are depicted in green, lost ones in red), and the markers indicate the age at the query arrival instants.

3.1 The QAoI metric

If the query arrival process is known in advance, we denote the query arrival times at the edge node by $t_{q,1}, t_{q,2}, \ldots$. We can then define the QAoI for the i-th query, denoted as $\tau(i)$ and given by:

$$\tau(i) = \Delta(t_{q,i}). \tag{D.4}$$

The EAoI metric proposed in [2] shares many similarities with QAoI, and indeed it also represents a pull-based system; however, it deals with concurrent queries and updates differently, allowing the server to wait until the update is over to respond to the query, while our formulation is stricter and enforces an order, with updates always arriving before queries. We then define the overall objective as minimizing the long-term expected QAoI, defined as

$$\tau_\infty = \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{i:t_{q,i}\leq T} \Delta(t_{q,i})\right]. \tag{D.5}$$

It is also possible to optimize QAoI without full knowledge of future query arrival times, as long as there is some information on the statistics of the process: in our model, the query process is represented by a finite Markov chain with a state space $\mathcal{S}_q$ and a transition matrix P_q. The query process is then in a (known) state at any time instant, and queries are generated if the state is in a predetermined subset $\mathcal{Q} \subseteq \mathcal{S}_q$.

Relating this to the use case example, the Markov chain represents the monitoring application: in the simplest case, it requests the sensor reading to the ground station periodically, but in general queries can have complex periodicities that can be modeled by a Markovian process. In most of the simple cases, we have $|\mathcal{Q}| = 1$, and the interval between two consecutive queries is Independent and Identically Distributed (IID). We

can then rewrite the long-term QAoI in the more general case with stochastic queries as the following:

$$\tau_\infty(s_q) = \limsup_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T-1} \frac{\Delta(t+1)}{T} \sum_{s'_q(t+1)\in\mathcal{Q}} P_q(s_q, s'_q; t+1)\right], \tag{D.6}$$

where $P_q(s_q, s'_q; t+1)$ is the probability of transitioning from state s_q to state s'_q after $t+1$ steps, specified by the (s_q, s'_q)-th entry in $(P_q)^{t+1}$. In this way, the QAoI is considered only in the instants in which a query is happening, i.e., when the Markov chain representing the query process is in a state in which a query arrives. It is also possible to consider normalizing the QAoI over the number of queries instead of the number of steps, but this simply leads to a constant multiplying factor as T tends to infinity.

If the query arrival process is memoryless, e.g., queries follow a Bernoulli process, the strategies to minimize AoI and QAoI are the same, as well as their distributions, as the query process transition probabilities are independent of the current state. The same is true when the query arrival process is much faster than the sensor, i.e., when there is a query in each time slot. The opposite extreme is the case with deterministic query arrivals, in which the transition matrix is deterministic: the query arrival instants are then known *a priori*, and the sensor can optimize its transmissions to minimize QAoI directly. The most obvious example of this is given by periodic queries, but the same holds for any deterministic process that is known to the sensor.

In general, the QAoI can be very different from the AoI in a given system, as well as the strategies for optimizing it. In this work, we present such a class of systems, arguing that an approach that takes the nature and possible periodicity of the monitoring process into account can fit more situations and result in better performance than standard AoI minimization. This difference is shown in a simple example in Fig. D.3, in which the three metrics lead to different solutions: while minimizing AoI leads to periodic transmissions, QAoI and EAoI lead the sensor to transmit just before queries. However, EAoI-oriented systems would transmit one slot earlier, so as to avoid delaying the response to the query.

3.2 Communication system model

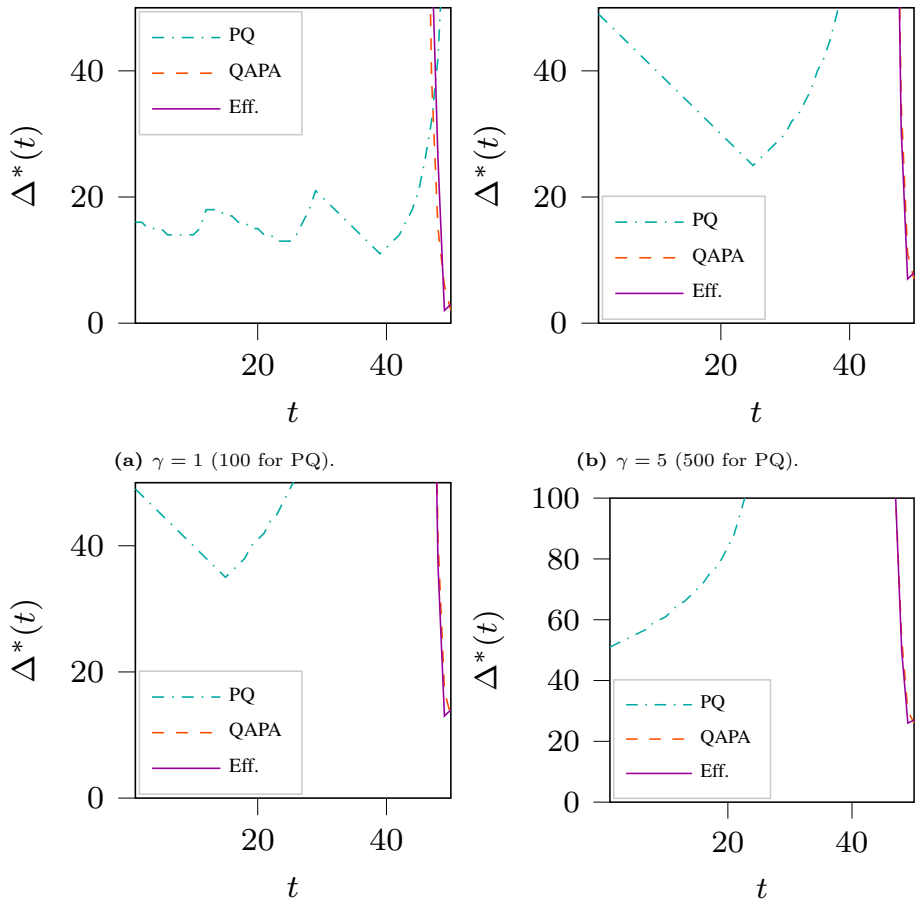
We now define the model of our pull-based communication scenario, in which a sensor needs to schedule transmissions over a link with limited availability. Independently, queries about the state of the sensor are generated, e.g., as part of a monitoring or control process. The objective of this work is to maximize the freshness of the information *at the query time*, while considering that the sensor is energy-constrained and needs to limit the number of transmissions to prolong its lifetime.

We assume that each update has a fixed size and is transmitted over a PEC with slotted time. The sensor and receiver are assumed to be synchronized, i.e., the sensor is informed when the last query arrived by the time of the next transmission window. In order to be as general as possible, we model the channel as a Markov chain with state space $\mathcal{S}_e$ and transition matrix P_e, as we did for the query arrival process. An error probability $\varepsilon(s)$ is associated to each state $s \in \mathcal{S}_e$, and packets are instantaneously acknowledged by the receiver, so the sensor knows if the last transmitted packet was erased or correctly received. The Markovian model can fit several practical scenarios, including periodic communications or a channel with random outages, and is often used in wireless communications. We also make the simplifying assumption that the sensor can know the state of the channel through beacon messages. This is a simplifying hypothesis, and is not always true in sensor networks. However, having to rely on older estimates of the state of the channel would not change the fundamental nature of the model, and would only increase the uncertainty of the transmission.

The simplest case we can examine is the constant and always available channel, in which $\varepsilon(t) = \varepsilon$. A slightly more complex example is a deterministic and periodic error process. This models links that are available in a cyclical manner with period T_e, like the orbital passes of LEO communication satellites. In this case, the error probability is 1 when no satellite is visible and constant during a pass. In our simulations, we limit ourselves to deterministic and periodic error probability processes, whose value is known by the transmitter, but the formulation and solution are general. In our simulations, we consider the four use cases presented in the introduction, which can be simply mapped to different Markov chains for the query and channel error processes: deterministic queries follow a Markov chain with T_q states and forced transmissions, as do predictable channels (with the constant channel as a special case with $T_e = 1$). Stochastic query processes and channels follow general finite Markov chains, and in this case, any foresighted action relies on knowledge of the transition probabilities of the Markov model.

4 Analytical example

We can now consider a simple example in which the difference between AoI and QAoI is clear, and in which we can derive the optimal strategies analytically. We assume that transmitting has a constant cost γ, that the channel has a constant error probability ε, and that the optimization is over a finite horizon T. The objective of the PQ system is then to minimize the average AoI during an episode, while the QAPA system will try to minimize the AoI at the query instant, i.e., in slot T. We also consider the EAoI metric from [2], which results in a system similar to QAPA.



(a) $\gamma = 1$ (100 for PQ).

(b) $\gamma = 5$ (500 for PQ).

(c) $\gamma = 10$ (1000 for PQ).

(d) $\gamma = 20$ (2000 for PQ).

Fig. D.4: Thresholds Δ^* for $T = 50$, $\varepsilon = 0.2$.

We then define a cost function for the two systems, which we call c:

$$c_{\mathrm{PQ}}(t) = \gamma a_t + \Delta(t); \tag{D.7}$$

$$c_{\mathrm{QAPA}}(t) = \gamma a_t + \delta(t-T)\Delta(T), \tag{D.8}$$

where $\delta(x)$ is the Dirac delta function. We can also consider a QAPA system which tries to minimize EAoI instead of QAoI, to show the difference between the two metrics:

$$c_{\mathrm{EAoI}}(t) = \gamma a_t + \delta(t-T)\left[\delta(a_t)\Delta(T-1) + (1-\delta(a_t)\Delta(T)\right]. \tag{D.9}$$

We then define the long-term cost $C(\tau) = \sum_{t=\tau}^{T} c(t), \tau \leq T$. The optimal policy π^*_{PQ} is then the one that minimizes $C_{\mathrm{PQ}}(\tau)$, and the same is true for the QAPA system. In the last step, with $t = T$, $c(T) = C(T)$, and both systems have the same expected cost if they start from the same conditions and take the same action:

$$\mathbb{E}[c(T)|a_t = a, \Delta(t-1) = d] = \begin{cases} \gamma + \varepsilon d + 1, & \text{if } a = 1; \\ d + 1, & \text{if } a = 0. \end{cases} \tag{D.10}$$

It is easy to see that transmitting in the last slot, i.e., setting $a_T = 1$, reduces the long-term cost if:

$$\Delta(T-1) > \frac{\gamma}{1-\varepsilon}. \tag{D.11}$$

We can then derive the optimal policies. In this case, any policy $\pi(d,t)$ depends only on the current age $\Delta(\tau-1) = d$ and the time slot index $\tau = t$. We only consider cases in which $\varepsilon > 0$, as the $\varepsilon = 0$ case is trivially optimized by transmitting only in the last slot before a query.

Theorem 1
The optimal policies π^*_{PQ} and π^*_{QAPA} are threshold policies, i.e., $\pi^*_{\mathrm{PQ}}(d,t) = 1 \Rightarrow \pi^*_{\mathrm{PQ}}(d+1,t) = 1$ and $\pi^*_{\mathrm{QAPA}}(d,t) = 1 \Rightarrow \pi^*_{\mathrm{QAPA}}(d+1,t) = 1$.

The proof of the theorem is given in the Appendix. The PQ policy is then defined by threshold values $\Delta^*_{\mathrm{PQ}}(t)$, defined as:

$$\Delta^*_{\mathrm{PQ}}(t) = \min\{d \in \mathbb{N} : \pi^*_{\mathrm{PQ}}(d,t) = 1\}. \tag{D.12}$$

The same holds for the QAPA policy. The threshold for transmission can be computed recursively: if we know π^*_{PQ} from time $\tau+1$ to the end of the episode, we can compute $\Delta^*_{\mathrm{PQ}}(\tau)$ using the following formula:

$$\begin{aligned} \Delta^*_{\mathrm{PQ}}(\tau) = \inf\Big\{ & d \in \mathbb{N} : d + \mathbb{E}[C(\tau+1)|\pi^*, \Delta(\tau) = d] \\ & > \frac{\gamma}{1-\varepsilon} + \mathbb{E}[C(\tau+1)|\pi^*, \Delta(\tau) = 0]\Big\}, \end{aligned} \tag{D.13}$$

where $\mathbb{E}[C(\tau+1)|\pi^*, \Delta(\tau) = d]$ is the expected long-term cost while following the optimal policy. On the other hand, $\Delta^*_{\text{QAPA}}(\tau)$ is given by:

$$\Delta^*_{\text{QAPA}}(\tau) = \inf\Big\{ d \in \mathbb{N} : \mathbb{E}[C(\tau+1)|\pi^*, \Delta(\tau) = d] > \frac{\gamma}{1-\varepsilon} + \mathbb{E}[C(\tau+1)|\pi^*, \Delta(\tau) = 0] \Big\}. \tag{D.14}$$

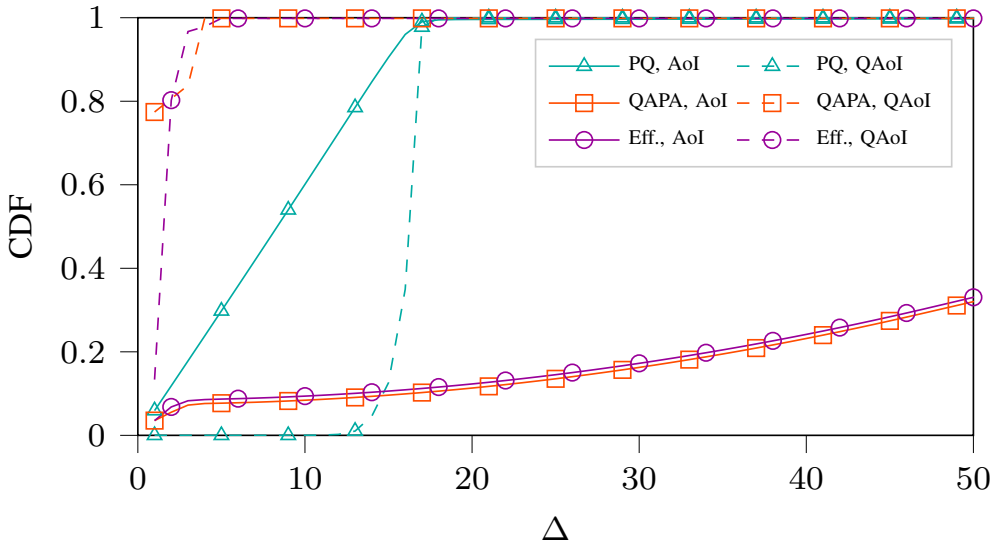
Naturally, we have $\Delta^*_{\text{PQ}}(T) = \Delta^*_{\text{QAPA}}(T) = \frac{\gamma}{1-\varepsilon}$, as given by (D.11).

Fig. D.4 shows how the threshold strategies work in a system with $\varepsilon = 0.2$: while the PQ system exhibits some periodic behavior, mostly due to the effect of the finite horizon, it tends to transmit less if the episode is close to the end, as the future reduction of the AoI is limited in time. On the other hand, the QAPA sensor transmits only in the last steps before the query, but increasing the value of γ makes it less convenient to transmit, and the sensor will only do so for a progressively higher expected QAoI, and only closer to the actual query instant. If we use EAoI as a metric for the QAPA system, the threshold is not monotonic anymore: the best moment to transmit is actually one step before the query, as transmitting in the same slot increases the delay in the query response. The transmission cost γ for PQ is higher than for both QAPA settings, because the overall cost is much higher, taking into account the AoI at every step and not just at the last step. The monotonicity of the strategy for QAoI, which holds over all our simulations in more complex systems, can make optimization easier, and a formulation without delayed responses can be simpler to implement for IoT gateways.

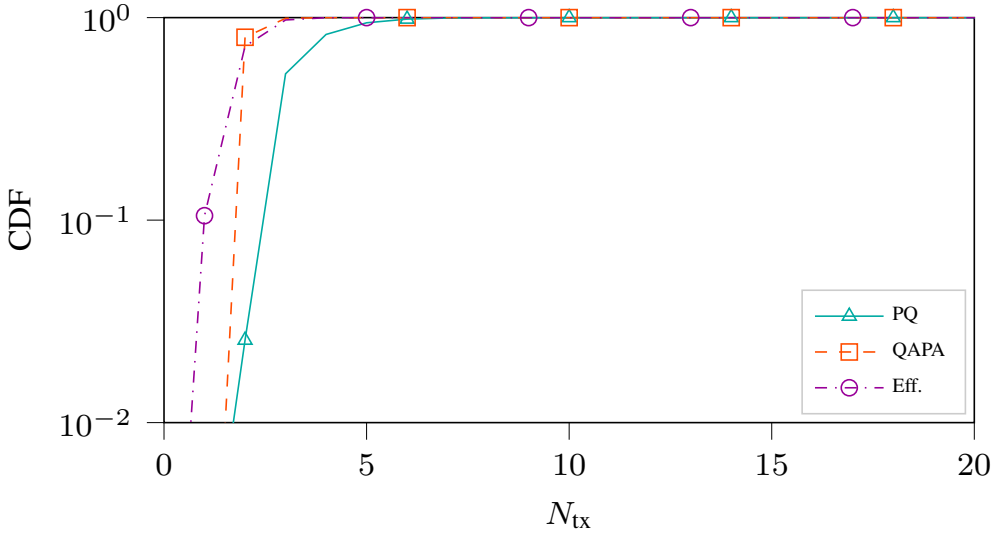
The performance of the systems is shown in Fig. D.5, which shows the CDF of the AoI and QAoI, along with the number of transmissions in each episode N_{tx}: as we can see, the QAPA strategy has a much better QAoI, even though it transmits slightly less. The EAoI-based system has a slightly higher QAoI than the QAoI-oriented system, as it tends to transmit slightly earlier: however, it also has slightly fewer transmission attempts on average. For both systems, this comes at the cost of the AoI, which is significantly higher than for the PQ system. As we will see in the next sections, this basic pattern holds even for much more complex systems.

5 MDP formulation and problem solution

In this section, we will consider the full problem, with a query process and energy dynamics. To understand the impact of the query process in the performance of a communication system, we will model the PQ and QAPA systems as MDPs, which we will then proceed to solve. A MDP is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a set of transition probabilities $p_a(s, s') = P(s_{t+1} = s' | a_t = a, s_t = s)$, and an instantaneous reward function $r(s, a, s')$, which represents the immediate reward when taking action a and transitioning from state s to state s'. The two systems, PQ and QAPA, can use the same state and action spaces, and only differ in the reward function that they use.



(a) CDF of the AoI and QAoI.



(b) CDF of N_{tx}.

Fig. D.5: Behavior of the PQ, QAPA, and EAoI systems with $T = 50$ and $\varepsilon = 0.2$, starting from a random age Δ_0 between 1 and 100.

This problem formulation is significantly more complex than the example from Sec. 4, making the direct derivation of strategies complex, but it can represent a more general class of communication systems.

To model the energy-constrained nature of the node, we use a *leaky bucket* model, as commonly done in the literature [32]: we consider a bucket of tokens, which is replenished by a process which can generate tokens independently at each step with probability μ_b. The node can only transmit a packet if there are tokens in the bucket, and each transmission consumes one token. This model can fit a general power consumption constraint on a battery-powered node, which should limit its number of transmissions in order to prolong its lifetime. Furthermore, it allows us to easily include the constraint in the MDP formulation as elaborated further in the next section.

Decisions are made by the sensor at every slot, as it can either keep silent or send a packet. Consequently, the action space is $\mathcal{A} = \{0, 1\}$. Both the PQ and the QAPA agents (i.e., sensors with two different objectives) need to know the current age $\Delta(t)$, as well as the state $s_e(t) \in \mathcal{S}_e$ of the error probability Markov process. Additionally, the agent should know the number of available tokens, $b(t)$, as it will influence its decision whether to transmit. If the number of tokens is 0, the sensor is blocked from transmitting until a token is generated. The tuple $(\Delta(t), s_e(t), b(t))$ is sufficient to represent the state in the PQ system, which does not require any knowledge of the query arrival process, while the QAPA system adds a fourth element to the state, i.e., the state of the query arrival process $s_q(t) \in \mathcal{S}_q$. As the PQ system can be studied as a special case of the QAPA system (with a single-state query arrival process), we adopt the wider definition for both systems to simplify the notation. We then define the state space as $\mathcal{S} = \mathbb{N}^2 \times \mathcal{S}_e \times \mathcal{S}_q$ and assume that the query arrival, token generation, and error probability processes are independent, examining each element of the state separately. The AoI increases by one between each slot unless the node decides to transmit and the packet is successfully received, with probability $p_s(t) = 1 - \varepsilon(s_e(t))$, in which case the AoI is reduced to one in the subsequent slot. The transition probabilities are thus described by

$$P(\Delta(t+1) = d | s_t, a_t) = \begin{cases} a_t p_s(t) & d = 1; \\ 1 - a_t p_s(t) & d = \Delta(t) + 1; \\ 0 & \text{otherwise,} \end{cases} \tag{D.15}$$

where a_t is the action at time t, which equals zero if the sensor is silent and one if it transmits. Secondly, the number of tokens in the next slot depends on whether a new token is generated and whether the sensor transmits, in which case it uses one token. The transition probability from $b(t)$ to $b(t+1)$ is:

$$P(b(t+1) = b + i | b(t), a_t) = \begin{cases} \mu_b & \text{if } i = 1 - a_t; \\ 1 - \mu_b & \text{if } i = -a_t; \\ 0 & \text{otherwise.} \end{cases} \tag{D.16}$$

The transition probabilities for the error probability and query arrival processes are defined by the matrices P_e and P_q, respectively. We assume that the query process is error-free, i.e., that the link between the ground station and the monitor is error-free.

We define two cost functions; one for the PQ system, which does not depend on the query instant and will be used as baseline, and one for the QAPA system, in which the cost is only considered when a query arrives. In the baseline PQ model, the cost is given by the AoI in any slot:

$$c_{\mathrm{PQ}}(s_t, a_t, s_{t+1}) = \Delta(t+1). \tag{D.17}$$

However, in the QAPA system, the cost is the AoI when a query arrives:

$$c_{\mathrm{QAPA}}(s_t, a_t, s_{t+1}) = \begin{cases} \Delta(t+1) & \text{if } s_q(t+1) \in \mathcal{Q}; \\ 0 & \text{otherwise.} \end{cases} \tag{D.18}$$

In both cases, the objective is to find a policy π^* that minimizes the long-term cost. In this initial work, we limit ourselves to consider the discounted case, which benefits from strong convergence guarantees, and defer the case with undiscounted costs to future work. In this case, at least one optimal policy is guaranteed to exist as a stationary deterministic decision rule [33], i.e. $\pi^* : \mathcal{S} \to \mathcal{A}$. Specifically, we solve

$$\pi^* = \arg\min_{\pi} \left[\sum_{t=0}^{\infty} \lambda^t \sum_{s_t \in \mathcal{S}} P(s_t | s_0, \pi) c(s_t, \pi(s_t)) \right], \tag{D.19}$$

where $\lambda < 1$ is the discount factor, and $c(s_t, a_t) = \mathbb{E}_{s_{t+1}}[c(s_t, a_t, s_{t+1}) | s_t, a_t]$ is the expected cost of taking action a_t in state s_t under either the PQ or QAPA model. Naturally, the energy constraint has a major impact on the cost, but it is implicit: as the sensor cannot transmit an update if it has no energy tokens, its age will increase, consequently increasing the long-term cost. In fact, we explicitly excluded a cost of transmission from the model, as the policy can already account for energy limitations by tuning its behavior and avoiding short-sighted choices that maximize the short-term reward while leading to long-term harm. The factor λ is crucial in this, as a higher discount value leads to a more foresighted policy.

5.1 Problem solution

We can now proceed to solve the MDP for the two systems we have defined using policy iteration, as described in [34, Ch. 4]. In order to apply the algorithm, we need to truncate the problem to a finite MDP. We do so by defining a maximum age $\Delta_{\max}$, a maximum query interval $T_{q,\max}$, and a token bucket size B: once the age, the query interval, or the number of tokens in the bucket reach the maximum, they cannot increase further. As long as the maximum values are sufficiently large, they are not reached

during normal operation and this simplification does not affect the optimal policies or their performance.

The policy iteration algorithm has two steps, policy evaluation and policy improvement, which are repeated until convergence. The algorithm is initialized with a policy function π^0 and a value function v_π^0, which are both set to all zeros. At each step n, we can use the current estimate of the value of a state to update the next estimate, getting the value $u^n(s, a, s')$, defined as:

$$u^n(s, a, s') = c(s, \pi^n(s), s') + \lambda v_\pi^n(s'). \tag{D.20}$$

The iterative steps are then:

1. The policy is evaluated using

$$v_\pi^{n+1}(s) = \sum_{s' \in \mathcal{S}} p(s'|s, \pi^n(s)) u^n(s, a, s'), \tag{D.21}$$

   for all s, where s is the current state, s' is the new state, a is the action, and c is the cost from either (D.17) or (D.18). The value function is an estimate of the long-term value that can be achieved in a given state using the policy.

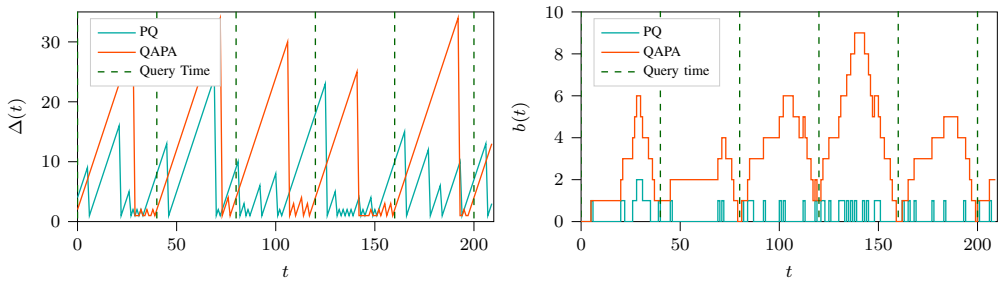
2. The policy is improved by choosing the action that maximizes the long-term value, i.e., minimizes the long-term cost:

$$\pi^{n+1}(s) = \arg\min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s'|s, a) u^{n+1}(s, a, s'). \tag{D.22}$$

Policy iteration is guaranteed to converge to the optimal policy [35] in finite-state MDPs with finite reward. The complexity of policy iteration in general is exponential in the number of states, making it particularly impractical for realistic problems. However, if the correct pivoting rule is adopted and the discount factor is constant in time, policy iteration was shown to be strongly polynomial in [36]:

$$N_{\text{it}} \leq \frac{|\mathcal{S}|^2(|\mathcal{A}| - 1)}{1 - \lambda} \log\left(\frac{|\mathcal{S}|^2}{1 - \lambda}\right). \tag{D.23}$$

Each iteration uses at most $O(|\mathcal{A}||\mathcal{S}|^2)$ operations, making the resulting bound polynomial in the size of the MDP. As mentioned above, we truncated the age and token bucket size to make the MDP finite, so the conditions to use the algorithm apply. The notation in the past two sections is summarized in Table D.1. In our case, in which there are only two actions, policy iteration then converges in $O\left(\frac{|\mathcal{S}|^2}{1-\lambda} \log\left(\frac{|\mathcal{S}|^2}{1-\lambda}\right)\right)$ iterations, which correspond to $O\left(\frac{2|\mathcal{S}|^4}{1-\lambda} \log\left(\frac{|\mathcal{S}|^2}{1-\lambda}\right)\right)$ steps.



(a) AoI over time for the two policies.

(b) Available tokens over time for the two policies.

Fig. D.6: AoI dynamics of the PQ and QAPA policies for $T_q = 40, \mu_b = 0.2, \varepsilon = 0.2$. The PQ policy generally has a lower AoI, but the QAPA policy minimizes the AoI at the query instants.

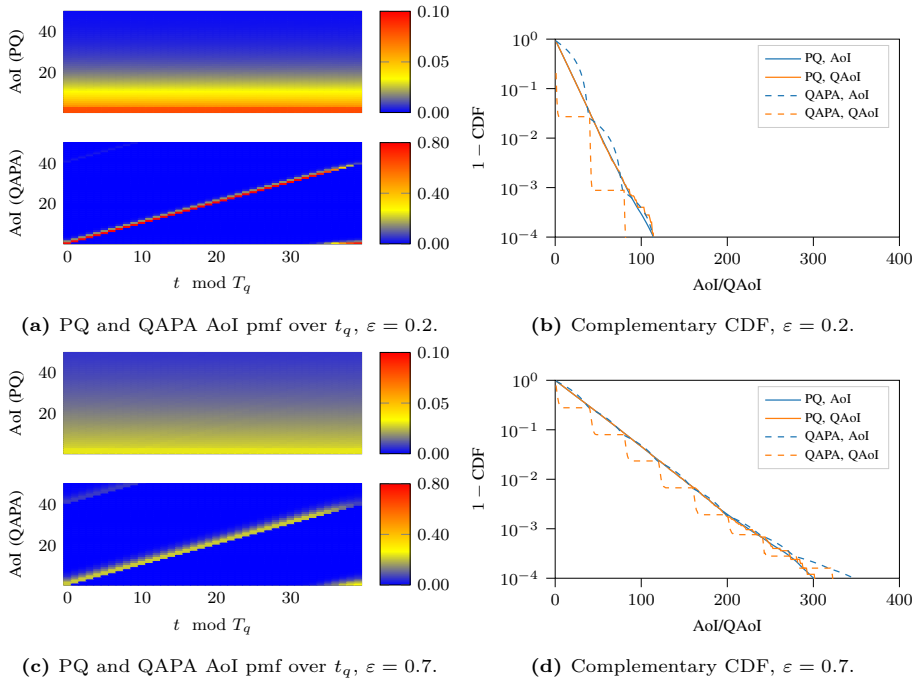
Naturally, policy iteration can suffer from the curse of dimensionality in more complex scenarios: as the number of states increases exponentially with the number of considered dimensions of the state, and policy iteration has itself an exponentially increasing complexity as the number of states increases, this solution is only practical for relatively small problems. In larger problems, learning-based techniques such as reinforcement learning can provide a far faster convergence, particularly when using deep reinforcement methods that can generalize experience [37]. These solutions have been successfully applied in the AoI optimization literature [25], and they can be applied directly to our MDP formulation, but we consider them as part of future work on this subject, solving some simpler examples with policy iteration.

6 Simulation settings and results

This section presents Monte Carlo evaluations of the policies obtained using the MDP described in Section 5. Although, the methods in Section 5 can be applied to any query process, throughout the evaluation we will consider queries that occur periodically, at a fixed time interval T_q. Furthermore, we truncate the MDP at a maximum age of $\Delta_{\max} = 100 \times T_q$ and a maximum token bucket size of $B = 10$, and we use a discount factor $\lambda = 0.95$. We use the term AoI to refer to the age at any time and QAoI for the age sampled at the query instants.

6.1 Periodic queries with constant error probability

We first consider the simplest scenario, in which the error probability is constant and the query arrival process is deterministic with period T_q. In this scenario, the error probability process only has one state, i.e., $|\mathcal{S}_e| = 1$, and the error probability is a constant value ε. The query arrival process is a deterministic Markov chain with T_q



(a) PQ and QAPA AoI pmf over t_q, $\varepsilon = 0.2$.

(b) Complementary CDF, $\varepsilon = 0.2$.

(c) PQ and QAPA AoI pmf over t_q, $\varepsilon = 0.7$.

(d) Complementary CDF, $\varepsilon = 0.7$.

Fig. D.7: AoI distributions and CCDFs for PQ and QAPA for $T_q = 40, \mu_b = 0.1$, and $\varepsilon = \{0.2, 0.7\}$.

states, with $\mathcal{S}_q = \{1, \ldots, T_q\}$. The transition probabilities are given by:

$$P_q(s_q, s_q') = \begin{cases} 1, & s_q < T_q \wedge s_q' = s_q + 1; \\ 1, & s_q = T_q \wedge s_q' = 1; \\ 0, & \text{otherwise.} \end{cases} \tag{D.24}$$

The subset of query states is given by $\mathcal{Q} = T_q$, i.e., the server sends a query only when the state reaches T_q, after which the state is reset to 1 and the counter starts increasing again. This Markov chain is equivalent to a periodic deterministic query process.

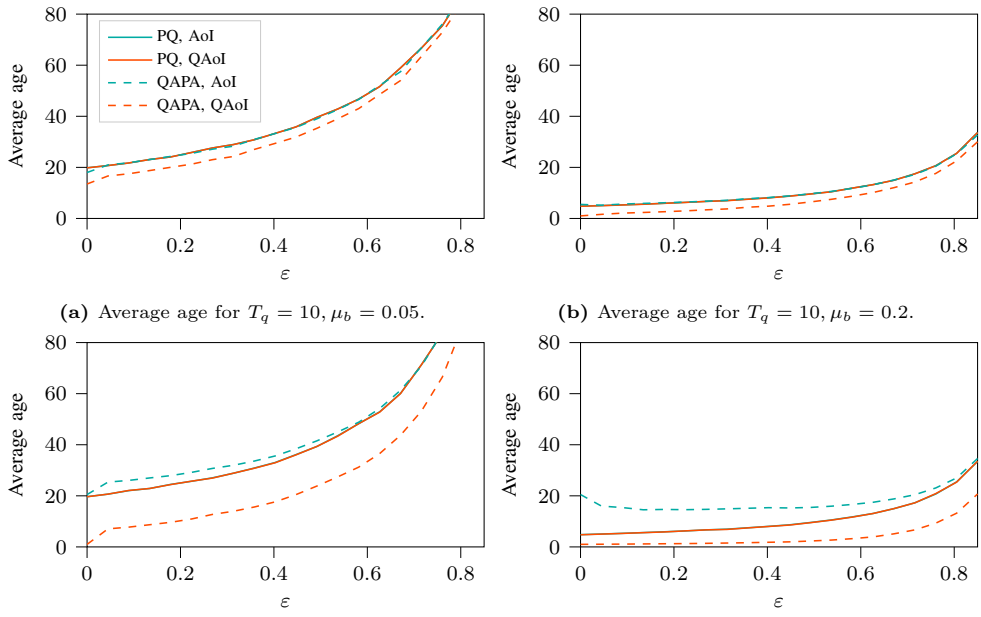
We start by exploring the temporal dynamics of the AoI process obtained using the PQ and the QAPA policies. Recall that PQ is optimized to achieve a low AoI independent of the query process, while QAPA minimizes the AoI at the query times, using cost functions (D.17) and (D.18), respectively. Fig. D.6a shows the AoI for queries occurring periodically every $T_q = 40$ time slots as indicated by the vertical lines, a packet error probability of $\varepsilon = 0.2$, and a token rate $\mu_b = 0.2$. It is seen that the PQ policy reduces the AoI approximately uniformly across time, while the QAPA policy consistently tries to reduce the AoI in the slots immediately prior to a query, so that the AoI is minimized when the query arrives. This is reflected in Fig. D.6b, which shows

that the QAPA policy accumulates energy when the next query is far in the future, unlike PQ. A consequence of this is that the QAPA policy generally has a slightly higher average AoI than the PQ policy, but the QAoI of the QAPA is significantly lower than that of the PQ policy.

The initial observations from Fig. D.6 can be confirmed by the distribution of the AoI as a function of the time since the last query, as illustrated in Fig. D.7. Figs. D.7a and D.7c show the probability mass function (pmf) of the AoI conditioned on various time instants $t \mod T_q$, while Figs. D.7b and D.7d show the CDF of the overall AoI and QAoI. We can immediately see the difference between the two policies from Fig. D.7a: the horizontal axis represents the time since the last query, while the colors represent the pmf of the AoI, whose domain is on the vertical axis. The AoI for the PQ policy does not depend on the time since the last query: the distribution is the same for all time instants, as can be seen by the fact that each horizontal line in the plot has exactly the same color. On the other hand, the QAPA policy shows a very different pattern: the AoI increases linearly as time passes, which indicates that the sensor does not send any packets in the first half of the interval, then sharply drops and stays very close to 0 in the final part of the interval. This behavior is consistent with what we would expect from a QAoI-oriented system, as sending packets long before the next query is basically a waste of energy, and transmissions are clustered just before the query instant. The resulting CDF in Fig. D.7b reveals, as expected, that the AoI and the QAoI are equivalent for the PQ policy, as the distribution is the same at any time instant. However, for the QAPA policy, the QAoI is significantly lower than the AoI, while the AoI is often larger than the PQ policy's. This is due to the fact that the QAoI is only measured at the query instants, at which the age of the QAPA policy is minimized. Due to the energy constraint, this comes at the cost of a generally higher age, causing a higher AoI measured at each time instant. Finally, the staircase appearance in the CDF is due to the fact that the queries happen periodically. If the queries were arriving at variable (but known in advance) intervals, then the QAPA would still have lower QAoI than the PQ query, but its CDF would have a different shape.

The same observations apply for the scenario with high error probability, $\varepsilon = 0.7$, shown in Figs. D.7c and D.7d. Although the AoI and QAoI are higher due to the high packet error rate, the applied policies are similar. It is interesting to note that there is a significant probability of having a QAoI higher than 40 (which corresponds to T_q) in Fig. D.7c, and that transmissions tend to be even more clustered towards the end of the query interval for the QAPA policy, indicating that the sensor will tend to save energy for future queries when it gets several tokens. Although there is a significant probability that the packet immediately prior to the query is lost, the AoI distribution at $t \mod T_q = 0$ is still concentrated close to one.

We now study how the average AoI and QAoI changes with the packet error probability ε for various choices of the parameters, shown in Fig. D.8. For all cases, the QAPA policy achieves the lowest QAoI, while the PQ policy achieves the lowest AoI.



(a) Average age for $T_q = 10, \mu_b = 0.05$.

(b) Average age for $T_q = 10, \mu_b = 0.2$.

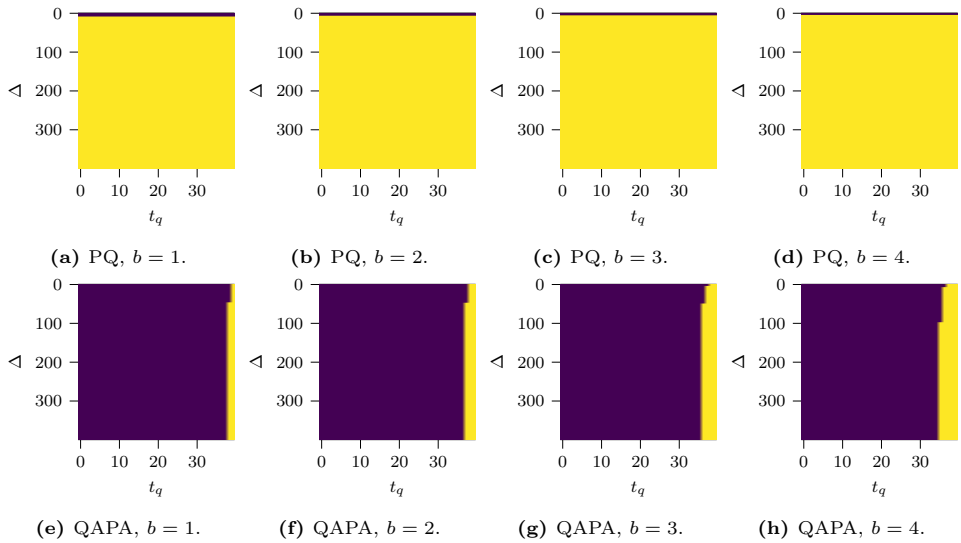
(c) Average age for $T_q = 40, \mu_b = 0.05$.

(d) Average age for $T_q = 40, \mu_b = 0.2$.

Fig. D.8: Average AoI and QAoI for the two systems for different values of T_q and μ_b.

When the query period, T_q, is low, the difference between AoI and QAoI is relatively small, as is the difference between the two policies. Intuitively, this is because the query instants, which are prioritized by the QAPA policy, are more frequent, making the two problems more similar.

As a result, awareness of the query arrival process becomes more important when queries are rare, i.e., when T_q is large: this is clear from the large gap between the average QAoI achieved by QAPA and by PQ in Fig. D.8c and Fig. D.8d. The plots on the left show the results for $\mu_b = 0.05$, i.e., when a new token is generated on average every 20 time slots. When $T_q = 10$ (see Fig. D.8a), the token period becomes a limiting factor, and both the AoI and QAoI are relatively high even for low values of ε. In particular, in the error-free case when $\varepsilon = 0$, the average QAoI cannot be lower than $(1+11)/2 = 6$, which is achieved by transmitting an update prior to every second query. Interestingly, the impact of the energy limit becomes less significant for the QAPA policy as the time between queries increases: by saving up tokens until right before the query, this policy can significantly reduce the QAoI, at the cost of a higher AoI. On the other hand, the PQ policy does not benefit from this increase, as it is oblivious of the query arrival frequency. When tokens are generated faster, at rate $\mu_b = 0.2$, as shown in the plots on the right side of the figure, the AoI and the QAoI are generally lower, since more frequent transmissions are allowed.



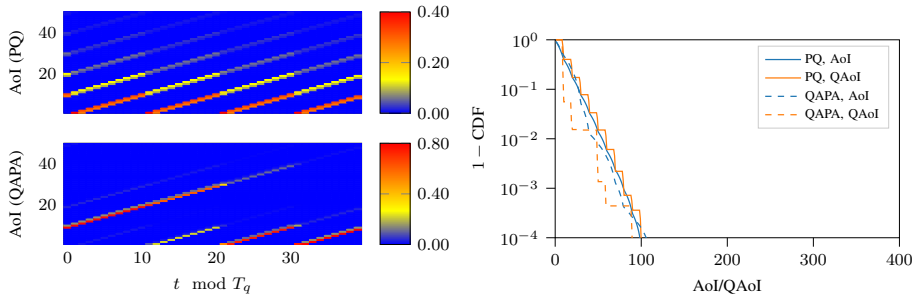
(a) PQ, $b = 1$. **(b)** PQ, $b = 2$. **(c)** PQ, $b = 3$. **(d)** PQ, $b = 4$.

(e) QAPA, $b = 1$. **(f)** QAPA, $b = 2$. **(g)** QAPA, $b = 3$. **(h)** QAPA, $b = 4$.

Fig. D.9: Strategies for the two systems with $\varepsilon = 0.2$. Yellow indicates a transmission, dark blue indicates no transmission.

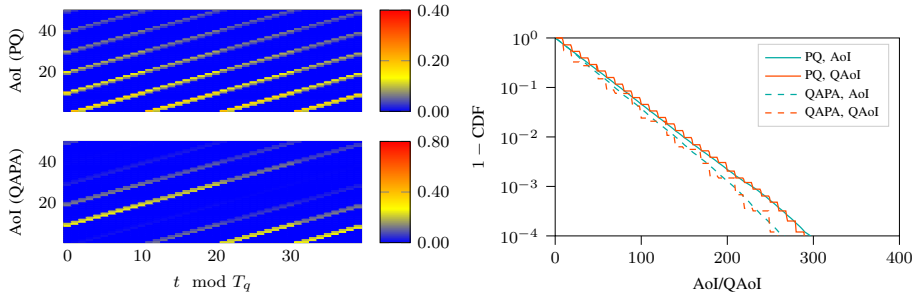
Finally, it is interesting to look at the strategies for the two systems, which are plotted in Fig. D.9. As expected, the optimal strategy for the PQ system does not depend on the queries, only on the current AoI. If the AoI is very low, the PQ system waits for a while to transmit, but this is limited to a few slots (8 if $b = 1$, and just 4 if $b = 4$). On the other hand, while the PQ strategy plots are horizontal, the QAPA plots are almost vertical, i.e., the importance of the AoI is very low, and the system decides almost only based on the time until the next query. Naturally, if b is low, it only tries to transmit in the very last possible slot, to reduce the QAoI, while more tokens allow it to start earlier and protect itself from errors. If the AoI is high enough, it might be worth it to start transmitting a little earlier and hope to get a new token in the next slot, as the small increase in the QAoI if only the packet transmission in the earlier slot is successful is offset by the large decrease if the new token arrives and only that transmission is successful. If we compare the QAPA strategies with the threshold-based ones derived for the simplified system in Fig. D.4, we see a significant similarity: the basic structure is still the same, although the precise values of the thresholds change. This result is encouraging for future attempts at deriving analytical threshold-based strategies for the full problem.

It is possible to imagine a simple threshold-based strategy that performs almost as well as the optimal strategy in this scenario: sending packets only if $b \geq T_q - t_q$ is a good approximation of the optimal strategy. Computing the threshold AoI over which it is convenient to send the first packet earlier is more complex, but still relatively easy



(a) PQ and QAPA AoI pmf over t_q, $\varepsilon = 0.2$.

(b) Complementary CDF, $\varepsilon = 0.2$.



(c) PQ and QAPA AoI pmf over t_q, $\varepsilon = 0.7$.

(d) Complementary CDF, $\varepsilon = 0.7$.

Fig. D.10: AoI distributions and CCDFs for PQ and QAPA for $T_q = 40, \mu_b = 0.1$ and $\varepsilon = \{0.2, 0.7\}$.

in this case. More complex cases, such as the ones we will examine below, have a much wider state space with a higher dimensionality, making strategies difficult to visualize and hand-design.

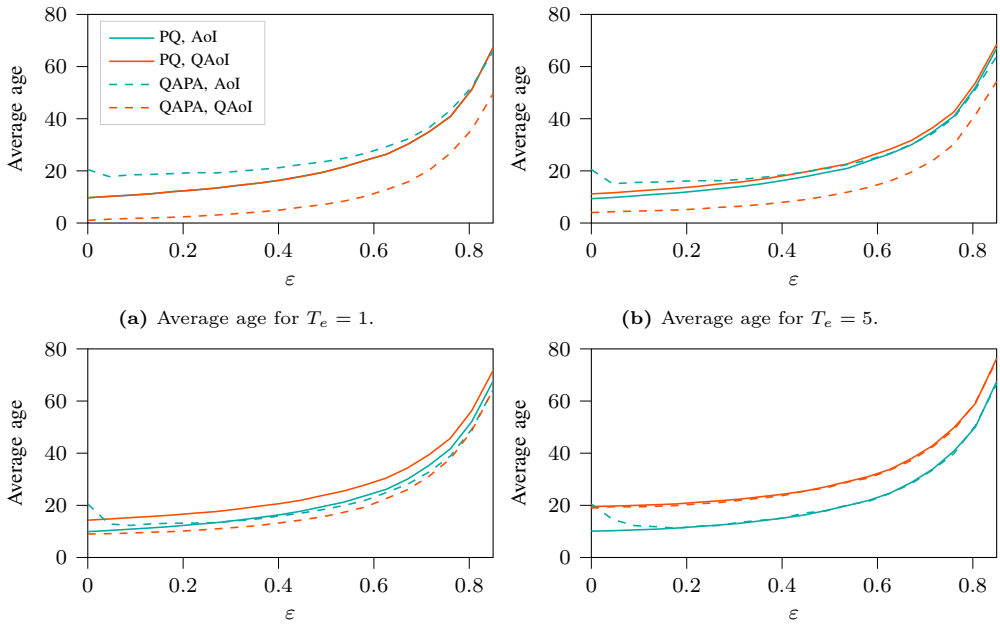
6.2 Periodic queries and error probability

We now analyze what happens when the error probability is not constant, but follows a periodic function. We consider the case of a LEO satellite connection which has limited availability due to constraints on the available constellation: connectivity is only available sporadically, depending on the periodic passes of the satellite over the transmitter node. In order to show the main trade-offs and represent a realistic case in which a LEO satellite passes over the transmitter at regular intervals, we consider a periodic error process with period T_e, in which the state space $\mathcal{S}_e = \{1, \dots, T_e\}$. The first two slots of each period are the only ones during which a transmission is possible, with an error probability ε_0, and correspond to the satellite pass. In all other slots, the transmission fails, as the transmitter is outside the satellite's coverage area. We then have $\varepsilon(1) = \varepsilon(2) = \varepsilon_0$, and $\varepsilon(s_e) = 1 \, \forall s_e \notin \{1, 2\}$. The transition probabilities for the error probability process are given by:

$$P_e(s_e, s'_e) = \begin{cases} 1, & s_e < T_e \wedge s'_e = s_e + 1; \\ 1, & s_e = T_e \wedge s'_e = 1; \\ 0, & \text{otherwise.} \end{cases} \tag{D.25}$$

We can analyze the behavior of the system as a function of the period T_e and the basic error probability ε_0. The query arrival process is defined as above, with transition probabilities given by (D.24). We first consider what happens in a simple case, setting $\mu_b = 0.1$, $T_e = 10$, and $T_q = 40$. Fig. D.10b and Fig. D.10d show how the PQ and QAPA system are restricted to the available transmission slots: outside of those slots, the AoI can only grow, resulting in the striped pattern on the plots. As expected, the QAPA system concentrates its effort in the transmission opportunities closer to a query, while the PQ system uses all available slots indiscriminately. This generates the difference in the QAoI seen in Fig. D.10b and Fig. D.10d: the QAPA system can maintain a lower QAoI, and the higher percentiles of the AoI, i.e., the tail of its distribution, are lower as well if the error probability is high.

We can then look at the effect of increasing the period of the satellite on the interplay between AoI and QAoI. We note here that we consider the worst possible scenario for the QAoI, i.e., the one in which the queries are synchronized with the satellite passes and each query arrives at the instant immediately before a satellite's pass. Fig. D.11 shows the difference in the average age for $T_e = 1$ (i.e., the previous scenario with constant error probability), $T_e = 5$, $T_e = 10$, and $T_e = 20$, considering $\mu_b = 0.05$ and $T_q = 40$. In all cases, the average AoI of the PQ process is similar: since the most important limiting factor is the energy constraint, the average age is about 20 slots in the error-free case



(a) Average age for $T_e = 1$.

(b) Average age for $T_e = 5$.

(c) Average age for $T_e = 10$.

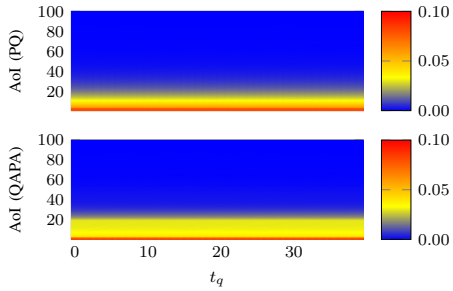
(d) Average age for $T_e = 20$.

Fig. D.11: Average AoI and QAoI for the two systems for different values of T_e, with $\mu_b = 0.1$ and $T_q = 40$.

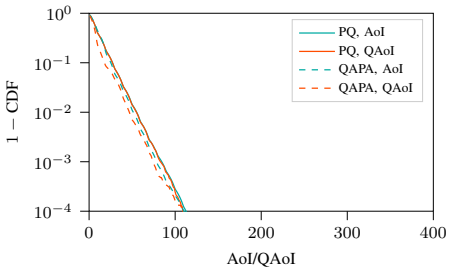
and follows a similar trend for all subfigures. By comparing Fig. D.11a and Fig. D.11d, it is clear that this is not true for QAoI: the effect of having transmissions only at the beginning of the period, at least $T_e - 2$ slots from the query, increases the average QAoI for the PQ process by approximately $T_e/2 - 1$ slots. As in the previous case, the QAPA system can improve the QAoI by paying a small cost in terms of AoI, but the difference between its QAoI and the PQ system's reduces as transmission opportunities become scarcer: by constraining the possible transmissions of the QAPA system to a few slots, we reduce the optimality gap of the traditional AoI maximization strategy. However, the difference between the two is still significant even for $T_e = 20$, as shown in Fig. D.11d. It is also interesting to see that AoI and QAoI are not the same in this case, even for the PQ policy: this is due to the fact that, as T_q is a multiple of T_e, queries always come just before a transmission opportunity, so the QAoI is always at least $T_e - 1$, while there is no such offset for AoI, which is measured in all slots.

6.3 Stochastic queries with periodic error probability

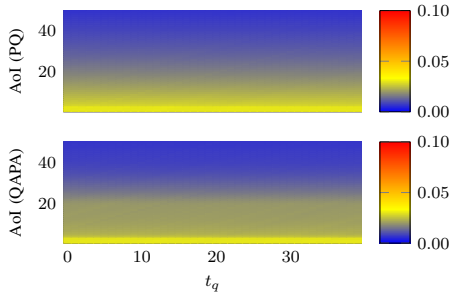
We now examine a more general case, in which queries arrive at stochastic IID intervals with a known distribution and transmission opportunities are limited by satellite passes.



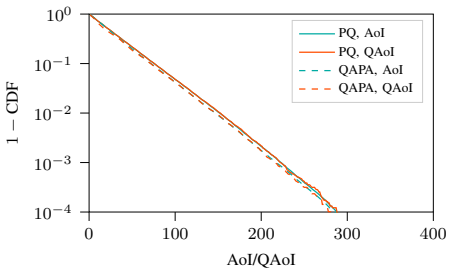
(a) PQ and QAPA AoI pmf over t_q, $\varepsilon = 0.2$.



(b) Complementary CDF, $\varepsilon = 0.2$.

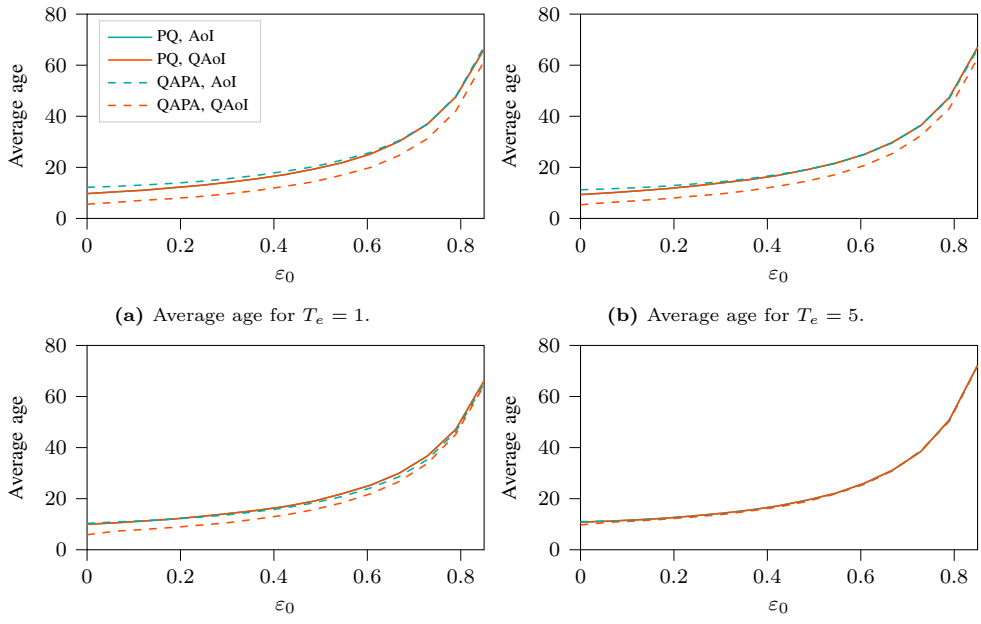


(c) PQ and QAPA AoI pmf over t_q, $\varepsilon = 0.7$.



(d) Complementary CDF, $\varepsilon = 0.7$.

Fig. D.12: AoI distributions and CCDFs for PQ and QAPA for $T_q = 40, \mu_b = 0.1, T_e = 10$, and $\varepsilon = \{0.2, 0.7\}$, with uniformly distributed query intervals over $\{21, 22, \ldots, 40\}$.



(a) Average age for $T_e = 1$.

(b) Average age for $T_e = 5$.

(c) Average age for $T_e = 10$.

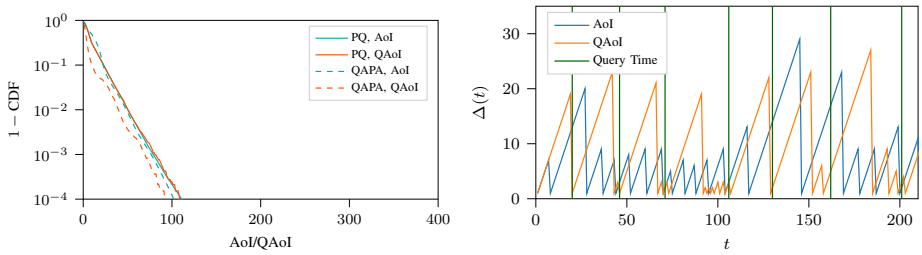
(d) Average age for $T_e = 20$.

Fig. D.13: Average AoI and QAoI for the two systems for different values of T_e, with $\mu_b = 0.1$, $T_q = 40$, and uniformly distributed query intervals over $\{21, 22, \ldots, 40\}$.

The error probability process is then defined as above, with transition probabilities given by (D.25). On the other hand, the queries are modeled to arrive with uniformly distributed inter-query times, i.e., $t_{q,i+1} - t_{q,i} \sim \mathcal{U}\left(\frac{T_q}{2} + 1, T_q\right)$. This is a worst-case scenario for the QAPA system: as queries can arrive at a random instant over a wide range of values, the transmitter needs to keep the AoI low almost at all times. We have $\mathcal{S}_q = \{1, \ldots, T_q\}$ with these non-zero transition probabilities:

$$P_q(s_q, s_q') = \begin{cases} 1, & s_q \leq \frac{T_q}{2} \wedge s_q' = s_q + 1; \\ 1 - \frac{1}{T_q - s_q + 1}, & \frac{T_q}{2} < s_q < T_q \wedge s_q' = s_q + 1; \\ \frac{1}{T_q - s_q + 1}, & s_q > \frac{T_q}{2} \wedge s_q' = 1. \end{cases} \tag{D.26}$$

Fig. D.12 shows that this is indeed a case in which knowledge of the query process is not critical: the age pmfs in Fig. D.12a and Fig. D.12c are almost the same for PQ and QAPA. In this case, the time since the last query has a limited value to the QAPA system, as it does not help much in predicting when a query will arrive. Consequently, the behavior of the QAPA system is much more similar to the PQ system's: the knowledge of the query arrival process statistics results in a very small gain, as the uniform



(a) Complementary CDF of the AoI and QAoI. **(b)** AoI over time for the two policies.

Fig. D.14: Behavior of the two systems with stochastic queries and a Gilbert-Elliott channel.

interval distribution gives a limited amount of information on future steps. However, setting a uniform distribution for the interval still implies some memory in the process, since, e.g., knowing that there has been no query for $T_q - 1$ steps implies that a query will arrive in the next step with probability 1. The PQ and QAPA strategies would be absolutely identical if the query process was Poisson, i.e., memoryless: if queries were to follow a memoryless process, any instant would be as valuable as any other in terms of future QAoI. This is evident in Fig. D.12b, which shows a negligible gain for the QAPA system in terms of QAoI, and even more in Fig. D.12d.

The analysis of the average AoI and QAoI as a function of the error probability ε_0, shown in Fig. D.13, shows that freedom of action and the precision of knowledge about the query arrival times are two factors that increase the gap between a naive PQ system and a query-aware QAPA one. This is intuitive, as limits to the available strategies can reduce gains, as can uncertainty about query arrival times. The more randomness is included in the system, and the more constrained the possible strategies become, the more QAoI looks exactly like AoI.

6.4 Stochastic queries with a Gilbert-Elliott channel

Finally, we consider another case, in which the channel does not have predefined transmission opportunities, but follows a stochastic Gilbert-Elliott [38, 39] model. In this model, we have two states, with $\varepsilon_1 = 0.2$ and $\varepsilon_2 = 0.7$. The transition probability matrix is given by:

$$P_e(s_e, s'_e) = \begin{pmatrix} 0.9 & 0.1 \\ 0.6 & 0.4 \end{pmatrix}. \tag{D.27}$$

On the other hand, the query process is given by the distribution in (D.26). In this case, both the channel and the query process are stochastic, and the amount of information available to the sensor is limited to the current state of each process and the transition probabilities.

Fig. D.14a shows the complementary CDF of the AoI and QAoI in this scenario: as this case is less restrictive than the one with periodic transmissions, in which a

successful update was only possible once every 5 or more slots, the QAPA strategy performs significantly better in terms of QAoI. Surprisingly, it also performs better in terms of the tail of the AoI, although not on average: this might be because of the choice to save energy by not transmitting in the first 20 slots after each query, which have a probability 0 of having another query. Fig. D.14b shows this behavior over time: while the PQ policy tends to uniformly transmit frequently, the QAPA policy lets the AoI grow when queries cannot arrive, then transmits more often to maintain the lowest possbile QAoI for a potential query.

7 Conclusions and future work

In this work, we have presented an optimization of QAoI, which takes the query arrival process and resource constraints on the communication into account. We showed that awareness of the query process can improve average and worst-case freshness in a variety of systems, modeling the single-source scheduling problem as an MDP and finding the analytical solution. As AoI does not consider the specific features of applications, but reduces the age of any packet at any time, it cannot incorporate this additional information. The awareness of the query process can significantly improve the freshness as perceived by several monitoring application, adapting the scheduling to only transmit when it is most useful and avoid useless updates.

This work is a first step in considering the requirements of time-sensitive monitoring applications in resource-constrained communication scenarios: we see several avenues of possible future work, such as including the value of updates in the scheduling problem as well as their timing. Furthermore, the extension of the problem to more complex systems with multiple sources and realistic channel access can be an interesting direction of research, as there are several scenarios with one or more monitoring applications that need information from multiple sensors. In these more complex scenarios, policy iteration would be too complex due to the curse of dimensionality, and we foresee the application of reinforcement learning methods to find the optimal strategy for minimizing QAoI.

A Appendix

In this Appendix, we prove that the threshold strategies defined in Sec. 4 are optimal. In order to prove Theorem 1, we first introduce the expected long-term cost $\Gamma(t, a, d)$:

$$\Gamma(t, a, d) = \mathbb{E}[C(t)|a_t = a, \Delta(t-1) = d]. \tag{D.28}$$

We can similarly define $\Gamma(t, \pi, d)$:

$$\Gamma(t, \pi, d) = \mathbb{E}[C(t)|\pi, \Delta(t-1) = d]. \tag{D.29}$$

We then introduce the following lemma.

Lemma 1
The expected long-term cost is a monotonically increasing function of the age Δ:

$$\Gamma(t, a, d) \leq \Gamma(t, a, d+1), \ \forall a, d, t. \tag{D.30}$$

Proof. We prove this lemma by backward induction in t, working from $t = T$ and going backward. If we consider the base case with $t = T$, we can see from (D.10) that the condition holds. We now assume that the condition is from $t+1$ to $t = T$. First, we consider the case with $a_t = 0$. We have:

$$\Gamma_{\mathrm{PQ}}(t, 0, d) = d + \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+1). \tag{D.31}$$

Since we know that $\Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+2) \geq \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+1)$ due to the assumption in the inductive step, and $d+1 > d$, the lemma is proven for $a_t = 0$. If we consider the case in which $a_t = 1$, we get:

$$\begin{aligned} \Gamma_{\mathrm{PQ}}(t, 1, d) =& 1 + (1-\varepsilon)\Gamma_{\mathrm{PQ}}(t+1, \pi^*, 1) \\ &+ \varepsilon(d + \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+1)). \end{aligned} \tag{D.32}$$

It is easy to see how this case also respects the condition, as d appears twice. All elements of the cost are then the same or higher for $\Delta(t-1) = d+1$ and the lemma is proven by induction. The same procedure can be repeated for QAPA, using either QAoI or EAoI as a metric. □

As the expected cost is a monotonically increasing function of the current age, we can immediately prove Theorem 1.

Proof of Theorem 1. As for Lemma 1, we can use backward induction starting from $t = T$ to prove the theorem. The base case is trivially true, as the decision is given by (D.11). We can now attempt to perform the inductive step *ad absurdum*, by assuming that the theorem is true from $t+1$ to T. Suppose that the optimal policy is not a threshold policy, i.e., $\exists d : \pi^*(d, t) = 1 \wedge \pi^*(d+1, t) = 0$. In this case, we know that, since $\pi^*(d, t) = 1$, we get:

$$\Gamma_{\mathrm{PQ}}(t, 1, d) < \Gamma_{\mathrm{PQ}}(t, 0, d). \tag{D.33}$$

We can then take the components of the reward:

$$\begin{aligned} 1 + d + \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+1) >& (1-\varepsilon)\Gamma_{\mathrm{PQ}}(t+1, \pi^*, 1) \\ &+1 + \gamma + \varepsilon(d + \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+1)), \end{aligned} \tag{D.34}$$

which yields:

$$d + \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+1) - \Gamma_{\mathrm{PQ}}(t+1, \pi^*, 1) > \frac{\gamma}{1-\varepsilon}. \tag{D.35}$$

On the other hand, since we assumed that $\pi^*(d+1,t) = 0$, we can follow the same procedure to get:

$$d + 1 + \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+2) - \Gamma_{\mathrm{PQ}}(t+1, \pi^*, 1) \leq \frac{\gamma}{1-\varepsilon}. \tag{D.36}$$

Since $C(t)$ is non-negative, and we know from Lemma 1 that $\Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+1) \leq \Gamma_{\mathrm{PQ}}(t+1, \pi^*, d+2)$, the results in (D.35) and (D.36) are in contradiction, and $\pi^*(d,t) = 1 \Rightarrow \pi^*(d+n,t) = 1, \forall n \in \mathbb{N}$. As before, the same procedure can be repeated for QAPA, which does not have the term d in the long-term cost, but still yields the same inequality. □

References

[1] 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.

[2] B. Yin, S. Zhang, Y. Cheng, L. X. Cai, Z. Jiang, S. Zhou, and Z. Niu, "Only those requested count: Proactive scheduling policies for minimizing effective age-of-information," in *Conf. on Computer Communications (INFOCOM)*. IEEE, Apr. 2019, pp. 109–117.

[3] B. Soret, S. Ravikanti, and P. Popovski, "Latency and timeliness in multi-hop satellite networks," in *Int. Conf. on Communications (ICC)*. IEEE, Jun. 2020.

[4] D. Li, S. Wu, Y. Wang, J. Jiao, and Q. Zhang, "Age-optimal HARQ design for freshness-critical satellite-IoT systems," *IEEE Internet of Things Journ.*, vol. 7, no. 3, pp. 2066–2076, Dec. 2019.

[5] J. Holm, A. E. Kalør, F. Chiariotti, B. Soret, S. K. Jensen, T. B. Pedersen, and P. Popovski, "Freshness on demand: Optimizing Age of Information for the query process," in *Int. Communications Conf. (ICC)*. IEEE, Jun. 2021.

[6] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Conf. on Computer Communications (INFOCOM)*. IEEE, Mar. 2012, pp. 2731–2735.

[7] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Age-optimal information updates in multihop networks," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2017, pp. 576–580.

[8] A. M. Bedewy, Y. Sun, and N. B. Shroff, "The age of information in multihop networks," *IEEE/ACM Trans. on Networking*, vol. 27, no. 3, pp. 1248–1257, Jun. 2019.

[9] B. Wang, S. Feng, and J. Yang, "To skip or to switch? minimizing age of information under link capacity constraint," in *19th Int. Worksh. on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, Jun. 2018.

[10] K. Chen and L. Huang, "Age-of-information in the presence of error," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jul. 2016, pp. 2579–2583.

[11] R. Devassy, G. Durisi, G. C. Ferrante, O. Simeone, and E. Uysal, "Reliable transmission of short packets through queues and noisy channels under latency and peak-age violation guarantees," *IEEE Journ. on Selected Areas in Communications*, vol. 37, no. 4, pp. 721–734, Feb. 2019.

[12] H. B. Beytur, S. Baghaee, and E. Uysal, "Measuring age of information on real-life connections," in *27th Signal Processing and Communications Applications Conf. (SIU)*. IEEE, Apr. 2019.

[13] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," *IEEE Trans. on Mobile Computing*, Dec. 2019.

[14] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. on Information Theory*, vol. 63, no. 11, pp. 7492–7508, Nov. 2017.

[15] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2015, pp. 3008–3012.

[16] R. D. Yates and S. K. Kaul, "Status updates over unreliable multiaccess channels," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2017, pp. 331–335.

[17] R. D. Yates and S. K. Kaul, "Age of information in uncoordinated unslotted updating," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2020, pp. 1759–1764.

[18] R. Talak, S. Karaman, and E. Modiano, "Distributed scheduling algorithms for optimizing information freshness in wireless networks," in *19th Int. Worksh. on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, Jun. 2018.

[19] X. Chen, K. Gatsis, H. Hassani, and S. S. Bidokhti, "Age of information in random access channels," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2020, pp. 1770–1775.

[20] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," *IEEE/ACM Trans. on Networking*, vol. 28, no. 1, pp. 15–28, Dec. 2019.

[21] B. T. Bacinoglu, Y. Sun, E. Uysal-Bivikoglu, and V. Mutlu, "Achieving the age-energy tradeoff with a finite-battery energy harvesting source," in *Int. Symp. on Information Theory (ISIT)*. IEEE, Jun. 2018, pp. 876–880.

[22] X. Wu, J. Yang, and J. Wu, "Optimal status update for age of information minimization with an energy harvesting source," *IEEE Trans. on Green Communications and Networking*, vol. 2, no. 1, pp. 193–204, Nov. 2017.

[23] Y. Gu, H. Chen, Y. Zhou, Y. Li, and B. Vucetic, "Timely status update in Internet of Things monitoring systems: An age-energy tradeoff," *IEEE Internet of Things Journ.*, vol. 6, no. 3, pp. 5324–5335, Feb. 2019.

[24] S. Feng and J. Yang, "Optimal status updating for an energy harvesting sensor with a noisy channel," in *Conf. on Computer Communications Worksh. (INFOCOM)*. IEEE, Apr. 2018, pp. 348–353.

[25] E. T. Ceran, D. Gündüz, and A. György, "Reinforcement learning to minimize age of information with an energy harvesting sensor with HARQ and sensing cost," in *Conf. on Computer Communications Worksh. (INFOCOM)*. IEEE, Apr. 2019, pp. 656–661.

[26] S. Zhang, H. Zhang, Z. Han, H. V. Poor, and L. Song, "Age of information in a cellular internet of UAVs: Sensing and communication trade-off design," *IEEE Trans. on Wireless Communications*, vol. 19, no. 10, pp. 6578–6592, Jun. 2020.

[27] Y. Sang, B. Li, and B. Ji, "The power of waiting for more than one response in minimizing the age-of-information," in *Global Communications Conf. (GLOBECOM)*. IEEE, Dec. 2017.

[28] F. Li, Y. Sang, Z. Liu, B. Li, H. Wu, and B. Ji, "Waiting but not aging: Optimizing information freshness under the pull model," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 465–478, 2020.

[29] M. Hatami, M. Jahandideh, M. Leinonen, and M. Codreanu, "Age-aware status update control for energy harvesting IoT sensors via reinforcement learning," in *31st Ann. Int. Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, Aug. 2020.

[30] M. Hatami, M. Leinonen, and M. Codreanu, "AoI minimization in status update control with energy harvesting sensors," *IEEE Trans. on Communications*, Sep. 2021.

[31] C. Xu, X. Wang, H. H. Yang, H. Sun, and T. Q. Quek, "AoI and energy consumption oriented dynamic status updating in caching enabled IoT networks," in *Conf. on Computer Communications Worksh. (INFOCOM)*. IEEE, Jul. 2020, pp. 710–715.

[32] V. Raghunathan, S. Ganeriwal, M. Srivastava, and C. Schurgers, "Energy efficient wireless packet scheduling and fair queuing," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 3, no. 1, pp. 3–23, Feb. 2004.

[33] R. Bellman, "A Markovian decision process," *Journ. of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, Jan. 1957.

[34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[35] R. A. Howard, *Dynamic programming and Markov processes.* John Wiley, 1960.

[36] Y. Ye, "The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate," *Mathematics of Operations Research*, vol. 36, no. 4, pp. 593–603, Nov. 2011.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[38] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journ.*, vol. 39, no. 5, pp. 1253–1265, Sep. 1960.

[39] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell System Technical Journ.*, vol. 42, no. 5, pp. 1977–1997, Sep. 1963.

Paper E

SENDAI: A Framework for Joint Reasoning About Sensor Data Acquisition and Sensor Data Analytics

Søren Kejser Jensen, Josefine Holm, Federico Chiariotti, Christian Thomsen, Anders Ellersgaard Kalør, Petar Popovski, Beatriz Soret and Torben Bach Pedersen

The paper has been submitted to the
Information and Computation Journal 2022.

The layout has been revised.

Abstract

Sensor networks are increasingly being deployed to monitor critical infrastructure. However, as the number of sensors being deployed increases, so does the amount of sensor data that must be transmitted, stored, and analyzed. Thus, a significant number of methods have been proposed to improve both sensor data acquisition and analytics. For example, for sensor data acquisition efficient sampling and transmission strategies have been proposed, and for sensor data analytics efficient compression and query processing methods have been proposed. However, the proposed strategies and methods generally focus exclusively on either sensor data acquisition or analytics and in this way limit the types of optimizations that can be performed. This paper proposes the Framework for joint Sensory Data Acquisition and Analysis (SENDAI), an integrated framework for reasoning about sensor data acquisition and analytics together. In order to motivate and design SENDAI, two very disparate industrial use cases are analyzed: (i) monitoring wind turbines and (ii) measuring utility consumption using smart meters. In addition, to demonstrate how SENDAI can be used to reason about sensor data acquisition and analytics together, it is shown how sensor data acquisition can be optimized to respond efficiently to a query workload.

1 Introduction

Sensor networks are increasingly deployed for tasks such as monitoring, automation, and fault-detection [1]. They generally transmit the collected data points to edge nodes with limited hardware, which then transmit the data points to significantly more powerful cloud nodes, e.g, hosted by one or more cloud providers like Amazon Web Services, Google Cloud, and Microsoft Azure. For example, a modern wind turbine contains thousands of high quality sensors with wired power and connectivity that transmit the collected data points to a central location in the wind turbine or the park before they are transmitted to the cloud. The sensors collect information about both the wind turbine itself and the weather, e.g., temperature, wind direction, and vibration. Another example is smart meters that provide detailed information about utility consumption, e.g., electricity and water. The smart meters wirelessly transmit the collected data points to edge nodes, and the edge nodes transmit the data points wirelessly to the cloud. For both of these use cases, the available hardware resources, e.g., bandwidth and storage, significantly limit the number of data points that can be collected due to the high upgrade cost. Specifically, for the wind turbine use case the main bottleneck is the bandwidth between the edge nodes and the cloud nodes, and for the smart meter use case the bandwidth between the sensors and the edge nodes is the main bottleneck. Thus, while many use cases would benefit from collecting additional data points, e.g., for optimizing the production of renewable energy from wind turbines, it is currently

infeasible.

Multiple different communities have proposed many methods for improving both *sensor data acquisition* and *sensor data analytics*. For example, the communication community has proposed a large number of efficient sampling and transmission strategies [2], while the database community has proposed a large number of efficient methods for storing and analyzing large quantities of sensor data [3–5]. Unfortunately, most sensor data acquisition strategies and methods do not take the requirements of complex sensor data analytics into account, while most methods developed for sensor data analytics do not consider sensor data acquisition. By taking a holistic approach that optimizes sensor data acquisition based on the requirements of the analytics that will be performed, the amount of sensor data to collect can be significantly reduced [6–8]. Thus, integrated methods for sensor data acquisition and analytics reduce the amount of bandwidth, storage, and computation required and enable the deployment of sensors at a previously unprecedented scale. Towards creating such integrated methods, this paper analyses two disparate real-life use cases, combines concepts from the communication and database community to propose the Framework for joint Sensory Data Acquisition and Analysis (SENDAI) which provides a holistic view of sensor data acquisition and analytics, and shows how SENDAI can be used to reason about sensor data acquisition based on the queries that will be executed. Thus, the paper makes the following contributions:

- (i) An analysis of sensor data acquisition and analytics for two very disparate real-life use cases: wind turbines and smart meters. This analysis is based on information provided by multiple collaborators from industry.
- (ii) The Framework for joint Sensory Data Acquisition and Analysis (SENDAI), a general framework derived from the analysis of the two use cases for reasoning about sensor data acquisition and analytics together.
- (iii) A demonstration of how the information in SENDAI can be used to reason about sensor data acquisition based on a query workload for the two very disparate use cases.

The rest of the paper is organized as follows: Section 2 presents a running example and describes concepts that are used throughout the paper. Section 3 contains the analysis of the wind turbine and smart meter use cases. Section 4 presents SENDAI while Section 5 describes how the information in SENDAI can be used to reason about sensor data acquisition for different use cases and query workloads. Section 6 describes related work, while Section 7 contains the conclusion and discusses future work.

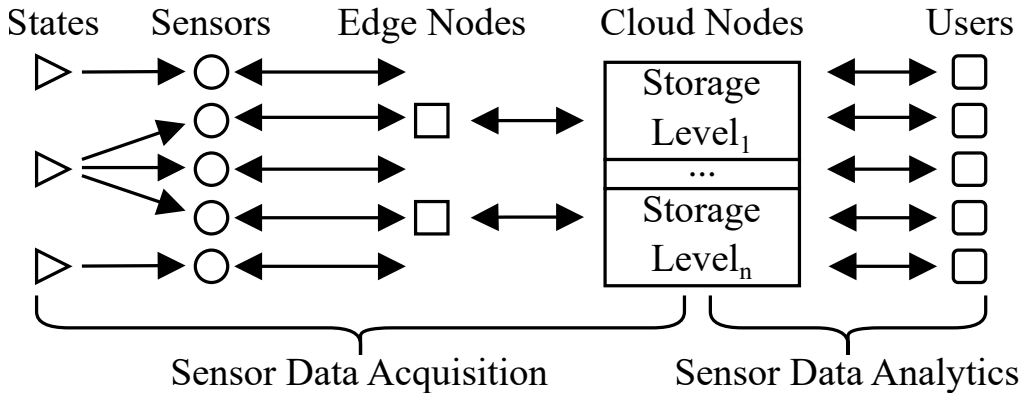


Fig. E.1: Generalized example of sensor data acquisition and analytics

2 Background

2.1 Running Example

A generalized example of sensor data acquisition and analytics which will be used throughout the paper is shown in Figure E.1. While generalized, the example is primarily based on the wind turbine and smart meter use cases mentioned in the introduction and which will be described in detail in Section 3. As the requirements and hardware used for sensor data acquisition and analytics differ significantly, the example is explicitly split into sensor data acquisition and analytics. In the example, sensor data acquisition consists of changing *States* that are being sampled by *Sensors* to produce data points:

Definition 1
Data Point: A *data point* $DP = (t, v)$ is pair where t is the time instant when the value $v \in \mathbb{R}$ was collected.

The data points are transmitted from the Sensors to *Edge Nodes*, and from the Edge Nodes to *Cloud Nodes*. Sensor data acquisition can either be *push-based* where the sender decides when to transmit data points to the receiver, *pull-based* where the receiver decides when to request data points from the sender, or *push-based with additional data points pulled if needed*. More details about sampling are provided in Section 2.2, while more details about transmission of data points are provided Section 2.3.

In the example, sensor data analytics consists of data points that are stored on Cloud Nodes with *tiered storage*. Thus, frequently accessed data points (hot data) may be stored in memory or on fast solid state drives, while data points that are rarely accessed (cold data) may be offloaded to hard disk drives or network attached storage.

Users may *query* the Cloud Nodes for specific data points from a set of Sensors within a time interval or an aggregate of the values from a set of Sensors within a time interval. The Cloud Nodes may request additional data points from the Sensors through the Edge Nodes if this is necessary to be able to answer the query. Alternatively, the Sensors, Edge Nodes, and Cloud Nodes may use *prediction*, i.e., using statistical or machine learning methods to estimate the values of lost historical data points or future data points. More details about prediction are provided Section 2.4.

2.2 Sampling

Sampling is the process of collecting data points from Sensors that monitor States as shown in Figure E.1. As stated sampling can either be push-based, pull-based, or push-based with additional data points pulled if needed. Sampling is an energy-consuming process and Sensors may have significant battery restrictions. Thus, energy consumption must be optimized for many use cases. Push-based sampling can be grouped into three fundamental categories based on energy consumption:

- **At-will Sampling** is the most energy-intensive, but also more precise, type of sampling: in this scheme, the Sensors are always awake, can sample the State at any point in time, and have free transmission scheduling [9]. The Sensor can then generate a sample at any time, and the sampling process can be optimized to maximize the accuracy.
- **Event-based Sampling** methods are triggered by an event [10], i.e., a significant change in a State with respect to the last value sampled from that State. This type of sampling is much more energy-efficient than At-will Sampling, as a simple comparison with a threshold can be performed by simple circuits without waking up the Sensor's main processing unit. However, it relies heavily on the threshold to determine when a significant change has occurred and thus trigger an event.
- **Periodic Sampling** is the simplest and most energy-efficient sampling method [11], as the Sensors measure the States at a fixed sampling interval instead of continuously or when an event occurs. Thus, the Sensors can remain in sleep mode between collecting each data point. Periodic sampling also makes it simpler to optimize the transmission of data points from the Sensors to the Edge Node. Since the data points are collected with a known sampling interval, the Edge Nodes can assign different transmission windows to the Sensors to avoid interference.

In general, there is a clear trade-off between the predictability of sampling times, which allows for a more efficient use of energy and communication resources, and the ability of the sampling method to adapt to the States' values. Periodic Sampling is preferable if energy and communication efficiency is required. However, it is also the least flexible method, as it does not consider the values of the States at all. On the

other hand, At-will Sampling is energy- and computation-intensive, and requires the that Sensors decides when and what to transmit which might cause network congestion and packet collisions. However, At-will Sampling can also react instantaneously to even minor changes in the values of the States, thus providing significantly higher flexibility.

As an alternative to the three categories of push-based sampling, pull-based sampling can be used [12]. As stated, with pull-based sampling the Edge Nodes decide when and which Sensors to request data points from. Generally, each Sensor then provides the most recent data point. Pull-based sampling can also be considered a version of Event-based Sampling where the Sensors are triggered by messages from the Edge Nodes instead of changes in the States [13]. As sampling is triggered by the Edge Nodes, pull-based sampling cannot adapt to the values of the States. However, it can still be very flexible, as the Edge Nodes can dynamically decide which Sensors to request data points from, e.g., by dynamically learning which Sensor to request data points from based on the queries that are being executed on the Cloud Nodes. In addition, as the transmissions are scheduled by the Edge Nodes, there is no interference between Sensors. However, the need to transmit requests to each Sensor can significantly reduce the amount of available bandwidth.

2.3 Transfer

The amount of data points that can be transmitted between a sender and a receiver is generally limited by the available bandwidth and the available transmission time. In addition, for battery-powered Sensors the amount of available energy and the energy used per transmission must also be taken into account. Thus, in this section, only the transmission of data points from Sensors to Edge Nodes is discussed. Generally, transmission of data points from the Sensors to the Edge Nodes is strongly dependent on the sampling and transmission methods used. For example, if the Sensors can only communicate with the Edge Nodes over a shared wireless *communication link*, two or more simultaneous transmissions from different Sensors will interfere with each other. Methods for transmitting data points can be categorized as either *Scheduled* or *Random Access*:

- In a push-based **Scheduled** transmission scheme, the Sensors are pre-configured such that only one Sensor transmits at any given time [14]. Naturally, as each Sensor has to wait for its turn before it can transmit the next data point, it cannot immediately inform the Edge Nodes about a significant change in the value of the State it is monitoring. Pull-based sampling can also be used with a Scheduled transmission scheme, as the requests from the Edge Nodes act as scheduling information by granting access to the communication link to the specified Sensor and thus prevents the other Sensors from interfering [15].
- In a **Random Access** transmission scheme, each individual Sensor decides when

to transmit. A Sensor can decide to transmit if the State it is monitoring has changed more than a threshold compared to the last transmitted data point, or the decision to transmit can be based on a decision algorithm. However, the complexity of the computations and decisions that can be made on the Sensors are generally limited by their very limited computational capabilities. Also, it is significantly harder to coordinate the transmissions when each individual Sensor decides when to transmit, so a Random Access transmission scheme cannot support as much traffic as a Scheduled transmission scheme without risking a significant number of packet collisions [16]. Thus, Random Access transmission schemes can support adaptive transmissions and usually have lower overheads [17] than a Scheduled transmission scheme, but they are always exposed to the risk of interference and, consequently, have a higher packet loss.

2.4 Prediction

Prediction of values for data points is used for both sensor data acquisition [18, 19] and analytics [20]. Prediction can be performed using statistical methods or machine learning models trained on the data points collected by the Sensors. Methods for prediction can be categorized based on if historical or future values are being predicted:

- The values of missing historical data points can be predicated using **Imputation** methods, e.g., if a data point was lost due to a packet collision. Various interpolation methods such as linear interpolation, polynomial interpolation, and spline interpolation [20], are commonly used for predicting the values of missing data points. Machine learning models have also been proposed, e.g., using matrix factorization [21].
- The values of future data points can be predicated using **Forecasting** methods. Thus, decisions may be made earlier than if the data points were sampled. A significant number of methods for predicting the values of future data points have been proposed. These includes both statistical methods such as ARMA and ARIMA [20], and machine learning models such as N-BEATS [22] and Transformer [23]. The error of forecasting is generally strongly dependent on the *Age of Information (AoI)* [24], i.e., the time elapsed from the last sampled data point to time when the value of a data point was predicted. If the State the Sensor measures changes linearly, the error will also increase linearly with the AoI. In the more general case of States that do not change linearly, it can be assumed that the expected error will still be a monotonically increasing function of the AoI.

The choice of sampling method significantly impacts prediction [10] as the sampling interval and the error of the values of the sampled data points affect the error of the prediction models. For sensor data acquisition, prediction may be used to reduce the

number of data points that must be transmitted by deploying equivalent prediction models on both the sender and the receiver [18, 19]. Thus, the sender need only transmit updated parameters for the prediction model on the receiver and data points whose values the prediction model on the sender cannot predict within an error bound. This can significantly reduce the amount of bandwidth and energy required as a result. However, ensuring that the values are within an error bound can only be guaranteed in a push-based scenario where the sender decides when and what to transmit. In a pull-based scenario, the receiver does not have access to the actual values and must decide when and what data points to request from the sender based only on its confidence in the predicted data points' values [18, 19]. For sensor data analytics, prediction can be used to backfill missing historical values so analysis methods that assume the data points are sampled using Periodic Sampling can be used, or to predict the values of future data points to take advantage of a future change in the monitored States.

3 Use Cases

3.1 Wind Turbine

The following use case description is generalized based on information provided by multiple wind turbine owners and manufacturers. A modern wind turbine is equipped with thousands of high-quality Sensors, with park operators often monitoring thousands of wind turbines and manufacturers monitoring tens of thousands of wind turbines. These high-quality Sensors are installed with wired power and connectivity. Thus, it is not necessary to minimize the amount of energy used by the Sensors. The Sensors are sampled at an approximately regular interval or an irregular interval, e.g., based on when the values change, and the data points are transmitted to a central location in the wind turbine or the park over a wired communication link. Thus, the Sensors do not store data locally. The wind turbine manufacturers have access to sensor data at a higher frequency than the owners, e.g., if an error is detected high-frequency sensor data may be sampled and stored on the turbine for diagnostics. The hardware used for sensor data acquisition in each wind turbine is comparable to a standard desktop computer at the time the wind turbine was built. The collected sensor data is transmitted to a data center over a wired communication link either continuously or at a regular interval, e.g., every two hours. While the local infrastructure owned by the wind turbine owners and manufacturers has been used in the past, the sensor data is now commonly stored and processed at a cloud provider. While some wind turbine owners and manufacturers use specialized Time Series Management Systems (TSMSs) [5], compressed binary files with a column-based layout such as Apache Parquet are more commonly used. For sensor data analytics a combination of proprietary, e.g., tools provided by the cloud provider, and open-source tools such as Apache Spark and Python with Pandas are commonly used. Due to the use of high-quality Sensors with wired connectivity, out-of-order,

missing, and invalid readings rarely occur. In addition, both out-of-order and invalid readings can be corrected using established methods. Thus, missing readings are the only inherent sources of error. The data points are augmented with metadata which facilitates multi-dimensional queries, and the collected data points are analyzed through a set of aggregate queries with the general structure shown in Listing E.1.

```
SELECT {aggregation of columns} FROM {table}
WHERE time >= {start time} AND time < {end time}
AND {optional checks on extra columns}
GROUP BY {time resolution}, {optional columns}
```

Listing E.1: Queries for the wind turbine use case

For this use case, the primary bottlenecks are the amounts of bandwidth and storage required. Wind turbine owners and manufacturers would prefer to sample most Sensors at 10–50Hz and to have vibration Sensors sampled at 44KHz. However, while the Sensors can be sampled at this rate, due to the enormous volume (i.e., size) and velocity (i.e., speed of the acquisition and processing) of sensor data this would produce, only simple aggregates are transmitted from the wind turbine to the cloud. Also, while the cloud provides access to practically infinite amounts of storage, the cost of storing high-frequency sensor data is prohibitively high. As only simple aggregates are collected, outliers and fluctuations that could indicate problems with the wind turbines are lost. Lossless compression is used to reduce both the bandwidth and storage requirements.

3.2 Smart Meter

The following is based on information provided by a manufacturer of smart meters, e.g., for measuring the consumption of electricity and water. Thus, these smart meters operate as Sensors that continuously collect data points, e.g., water consumption in a private household. The collected data points are transmitted wirelessly from the smart meters to Edge Nodes at an interval as low as every five minutes. However, if an exceptional event such as a breach or a reverse flow is detected, information about the event is transmitted to the Edge Node immediately (i.e., within milliseconds). However, the specific transmission time is randomized to ensure the communication link is not flooded if multiple meters detect the event at the same time. Despite the use of randomization, the risk of interference increases significantly as the number of smart meters deployed in close proximity becomes sufficiently high. The Edge Nodes transmit the data points to the cloud over a wireless communication link provided by a mobile network operator. The total amount of data points and events stored in the cloud is unknown but significant. Various forms of data analytics are performed on the collected data, e.g, detection of leaks in water distribution networks.

The bottleneck for this use case is almost always transmission of data points from the Sensors to the Edge Nodes, due to the large number of smart meters deployed in close proximity. It is generally a requirement that the data points for a day can be accessed by the User on the morning of the following day, and that a 99.6%–99.8% end-to-end success rate is provided for the transmission of data points and events. However, while the error rate for the transmission of data points from the Edge Nodes to the cloud is unknown, it is known to be non-zero.

4 Framework for joint Sensory Data Acquisition and Analysis (SENDAI)

4.1 Overview

SENDAI is designed to enable joint reasoning about sensor data acquisition and analytics and is primarily based on the two use cases described in Section 3. Thus, SENDAI extends the generalized example for sensor data acquisition and analytics shown in Figure E.1. To simplify the description of SENDAI, summarized content from Section 2.1 is purposely reused in this section. SENDAI is split into two logical sets of *Parts*: sensor data acquisition and analytics. For sensor data acquisition, the Parts are States, Sensors, Edge Nodes, and Cloud Nodes; and for sensor data analytics, the Parts are Cloud Nodes and Users. The order of the Parts is assumed to be static, however, some use cases may only require a subset of the Parts. For example, if the Sensors transmit data points directly to the Cloud Nodes. Sensor data acquisition is assumed to be either push-based or pull-based. Additional data points can be pulled if needed for a query and the available bandwidth allow it. The States are monitored by the Sensors and they may change *discretely* or *continuously*. However, as the States are sampled, the Sensors always produce discrete data points, and the ordered data points from each Sensor is a *time series*:

Definition 2
Time Series: A *time series* TS is a sequence of data points ordered by time in increasing order $TS = \langle (t_1, v_1), (t_2, v_2), \ldots \rangle$. For each data point $DP_i = (t_i, v_i)$, $1 \leq i$, the timestamp t_i represents when the value $v_i \in \mathbb{R}$ was recorded. A time series $TS = \langle (t_1, v_1), \ldots, (t_n, v_n) \rangle$ with a fixed number of n data points is a *bounded time series*.

Definition 2 is a definition from [25]. Due to Definition 2, it is assumed that all time series are univariate and ordered. The communication links used for transmitting the data points have an associated bandwidth and there is a probability that data points are lost. The *transmission format* is any representation from which the data points can be reconstructed. Thus, *compression* may be applied. To further reduce the bandwidth

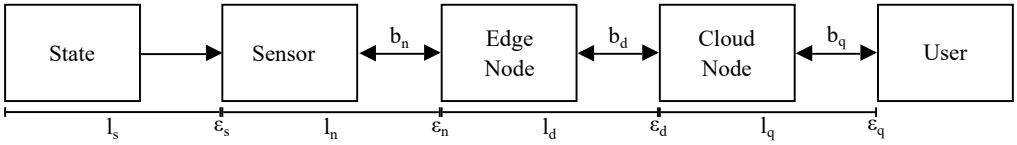


Fig. E.2: Overview of the separate metrics and bandwidth between Parts

required or recover lost data points, a Part may perform prediction, i.e., using statistical or machine learning methods to estimate the values of historical or future data points. Tiered storage allows hot data to be stored on fast storage, while cold data can be offloaded to slow storage. *Stochastic*, *periodical*, and *one-time* queries from the Users can be executed by the Cloud Nodes with additional data points collected from the Edge Nodes or Sensors if required. The queries are assumed to request data points from a set of Sensors for a specific time interval. If no time interval is given, the time interval is assumed to be from the beginning of time until the query is issued by the User.

The quality of a query result RS is defined by the time from the User has issued the Query until the User has fully received RS and the error of RS compared to a query result computed using values from the actual States. Thus, SENDAI reasons about sensor data acquisition and analytics based on the following two metrics: the *latency* l of RS and the *error* ε of RS. In most cases, the actual values of the States are unknown as only the collected data points are available. While latency can be measured without the actual values of the State, this is not possible for the error. Thus, SENDAI uses statistical bounds and probabilistic guarantees, e.g., guaranteeing that the 99th percentile of the error is below a threshold value. However, latency and error are not part of Definition 1 data point, so a *segment* is defined as:

Definition 3
Segment: A *segment* $S = (\langle (t_1, v_1), \ldots, (t_n, v_n) \rangle, dl_v, \varepsilon_v)$ is a bounded time series where $n \geq 1$; dl_v is the timestamp of the latest data point in S, if it was sampled, or the time it was predicted; and ε_v is the error of the values in S compared to the values of the actual State according to an error function.

A segment is purposely defined to support both error functions that compute the error for each data point and error functions that compute error for a bounded sequence of data points. Most error functions can be used for continuous values and any error function for discrete values can be used, but the complexity of the statistical evaluation naturally depends on the error function used. For the remainder of this paper, we limit the discussion to linear functions of the State, which have some properties that make the analysis simpler. However, SENDAI can be extended to more general classes of error functions with some modeling effort. When optimizing the latency and error of query results, the available resources, such as the amount of energy, bandwidth, and the amount of storage available to each Part, can be taken into consideration. As it is

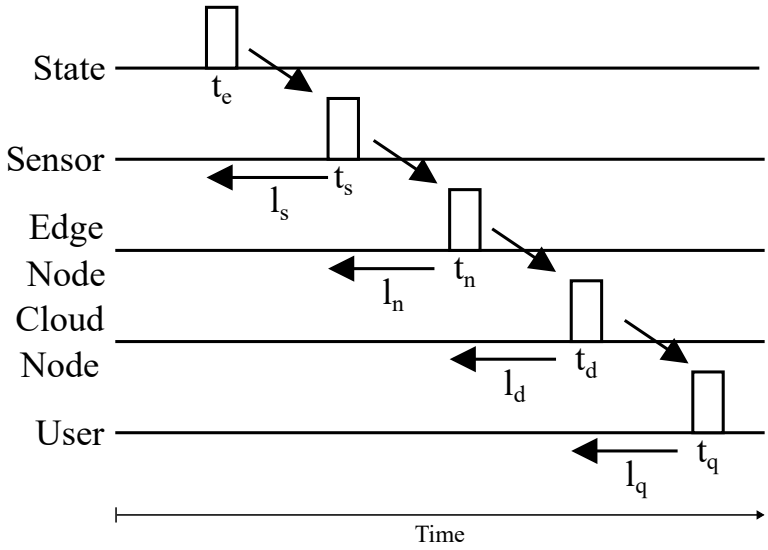


Fig. E.3: The latency added by each Part for a query that requires the State to be sampled

a limitation for both use cases, the bandwidth of the communication links is taken into consideration. Both latency and error are aggregates of separate metrics for each Part. These separate metrics and the bandwidth for the communication link between each Part are shown in Figure E.2. By assigning latency, error, and bandwidth to each Part it is simple to add and remove Parts, if necessary. Intuitively, the latency of RS is the time between the instant in which the User transmits a query to the Cloud Nodes to the instant when RS is returned. Thus, l is highly dependent on which Part can provide the data points required to compute RS when the query is transmitted. For example, if the value of a State at t_e must be sampled by a Sensor to compute RS, l is the latency added by all of the Parts as shown in Figure E.3. The latency added by each Part is defined as:

Definition 4
Latency: The *latency* l_i for a Part is $t_2 - t_1$ where t_1 is the time when the State changed or a segment was received at the previous Part, and t_2 is the time when the current Part received the same segment or a query result including that segment.

Thus, l_s is the time from the instant a State changes to when a Sensor has sampled that State, l_n is the time from the instant the Sensor has sampled the State to an Edge Node receives the segment, l_d is the time from that Edge Node receives the segment to the instant when a Cloud Node receives the segment, l_q is the time from that Cloud Node receives the segment to the User has fully received the query result that includes

that segment, and $l = l_s + l_n + l_d + l_q$. Sensors, Edge Nodes, and Cloud Nodes may buffer data points before transmitting them over a communication link and thus increase the latency. However, exactly how each Part adds latency is not explicitly modeled in SENDAI as the latency from buffering in a Part is accounted for by the total latency for that Part. The error of RS depends on the error of the segments used in its computation. The error added by each Part is defined as:

Definition 5
Error: The *error* ε_i for a Part is the difference $f_\varepsilon(S_1) - f_\varepsilon(S_2)$, where f_ε is a linear function, while S_1 and S_2 are segments that contain data points for the same time series and time interval, S_1 is received from the previous Part and S_2 is transmitted by the current Part.

For example, if we define the simplest possible $f_\varepsilon(S) = S$, and a Sensor measured $S_1 = 10$, but an Edge Node uses lossy compression so $S_2 = 12$, the error added by the Edge Node would be $\varepsilon_e = f(S_1) - f(S_2) = 2$. The total error ε_v of a segment S is the upper bound for the error added by the Part S is currently stored at, and all previous Parts. The actual error may be lower than ε_v, e.g., if multiple Parts use lossy compression and some overestimate S's values while others underestimate them, i.e., if the errors compensate each other. For multiple errors to be aggregated the following properties must hold:

Property 1
Additivity: If a query request is a linear function of the system State, error is additive: in a chain of segments from 1 to K, each with additional error ε_k with respect to its predecessor $k-1$, the total error is given by $\sum_{k=1}^{K} \varepsilon_k$.

This property is particularly useful if the error is normally distributed, as the statistics can be easily computed. In this case, it is possible to deal with correlations in the errors at different steps:

Property 2
Gaussian Error Compounding: If the errors ε_1 and ε_2 are normally distributed, with zero mean and variances σ_1^2 and σ_2^2, respectively, and have correlation ρ, the overall error ε_{1+2} is given by:

$$\varepsilon_{1+2} \sim \mathcal{N}\left(0, \sigma_1^2 + \sigma_2^2 + 2\rho\sigma_1\sigma_2\right). \tag{E.1}$$

In SENDAI, ε_s is the difference between the actual value and the segment sampled by a Sensor, ε_n is the difference between the segment transmitted by that Sensor and the segment transmitted by an Edge Node, ε_d is the difference between the segment transmitted by that Edge Node and the segment stored by a Cloud Node, ε_q is the difference between a query result computed from the segments stored on that Cloud Node and the query result transmitted by the Cloud Node, and the total error $\varepsilon =$

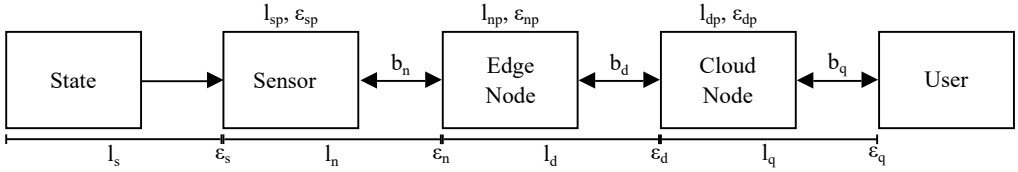


Fig. E.4: Overview of the separate metrics and bandwidths between Parts with prediction

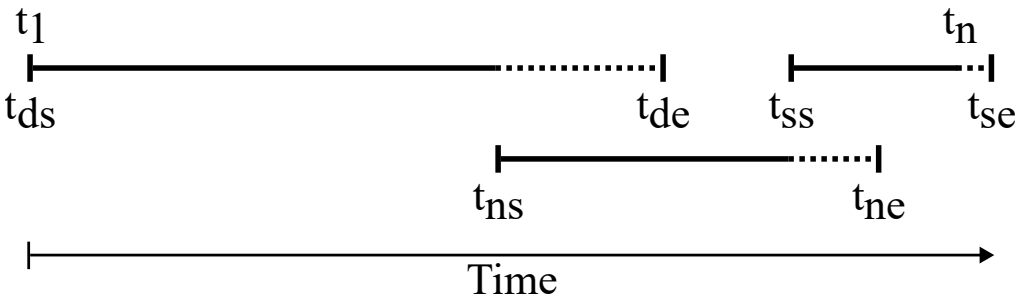


Fig. E.5: Example of Parts providing data points for overlapping time intervals using prediction. Full lines are time intervals for which sampled and predicted segments can be provided, while dotted lines are time intervals for which only predicted segments can be provided.

$\varepsilon_s + \varepsilon_n + \varepsilon_d + \varepsilon_q$ due to the additive property. In general, ε_s is caused by inaccuracies in the Sensors' measurements, the use of lossy compression, or the use of prediction. ε_n and ε_d are caused by the use of lossy compression or prediction. If the error is due to prediction, it will be proportional to the AoI of the segments used for the prediction, i.e., having more recent segments generally lead to better prediction. ε_q are caused by the use of prediction or Approximate Query Processing (AQP) techniques, e.g., reading only a sub-set of the available segments from disk when answering a query.

4.2 Prediction

As described above and in Section 2, prediction can be used to reduce the bandwidth required, reduce latency, or recover lost segments at the cost of additional error. Thus, both l and ε is affected by prediction. For example, if a segment is lost during transmission, it may be re-transmitted, which causes l to increase but lets ε remain constant. However, if prediction is used to avoid re-transmitting the segment by predicting the values it contained, ε generally increases while l decreases. Thus, prediction presents a different trade-off between the two metrics compared to sampling. Thus, the amount each Part contributes to l and ε depends on whether sampled segments or prediction was used. In SENDAI, both the latency and error of each Part when using prediction is modelled by assigning an additional latency and error metric to the Sensors (l_{sp} and ε_{sp}),

Edge Nodes (l_{np} and ε_{np}), and Cloud Nodes (l_{dp} and ε_{dp}) as shown in Figure E.4. These latency metrics are the amount of time required to predict the values of a segment, and the error metrics are error of the segment's values as defined Definition 5. The error for each prediction model can be modeled as a statistical variable, with the same properties as above, and can be compounded with other errors if it is Gaussian and the function of the State is linear. The time interval for which each Part can perform prediction with the specified error is modeled by associating each Part with a time interval for which that Part can provide segments with values within its sampling and prediction error. These segments can contain either sampled or predicted values. If prediction is used for forecasting, the time interval for which a Part can provide segments overlaps with the previous Part. An example is shown in Figure E.5. For the time interval from t_{ds} to t_{de}, the Cloud Nodes can provide segments with values within ε_d and ε_{dp}, while the Edge Nodes can provide segments with values within ε_n and ε_{np} from t_{ns} to t_{ne}, and the Sensors can provide segments with values within ε_s and ε_{sp} from t_{ss} to t_{se}. All three Parts use prediction in the form of forecasting to provide segments with more recent values than they have received from previous Parts, or in the case of the Sensor sampled from the State, as t_n is the current time. The Cloud Nodes forecast values within ε_{dp} from t_{ns} to t_{de}, the Edge Nodes forecast values within ε_{np} from t_{ss} to t_{ne}, and the Sensors forecast values within ε_{sp} from t_n to t_{se}.

4.3 Query Processing

To reason about query processing in SENDAI, the relational data model [26, 27] and relational algebra are used [28]. Thus, it is assumed that the time series are stored on the Cloud Nodes as segments and that Users execute queries against the relational TS_R as shown in Table E.1. Like the transmission format, the *physical storage format* used for the segments is any representation from which the segments can be reconstructed. In TS_R the data points from different time series are uniquely associated with a time series using a time series identifier. So a *data point with identifier* is defined as:

Definition 6
Data Point with Identifier: A *data point with identifier* $DP_i = (i, t, v)$ is triple where i is the identifier of the time series to which DP_i belong, and t is the time instant when the value $v \in \mathbb{R}$ was collected.

And a query is defined as:

Definition 7
Query: A *query* is a four-tuple $Q = (E, \varepsilon_r, ql_r, \xi)$ where E is a relational algebra expression, ε_r is the maximum acceptable error with a default value of 0, ql_r is the maximum acceptable query latency with a default value of ∞, and ξ is a stochastic arrival process that determines when Q is executed. E is either $\Pi(TS_R)$ or $\Pi(\sigma(TS_R))$ where σ is selection and Π is generalized projection.

Table E.1: TS_R with *Time_series_ID* and *Sample_Time* as the primary key

Time_series_ID (PK)	Sample_Time (PK)	Value
1	1990–05–01 12:00:00.000	13.45
1	1990–05–01 12:00:00.000	13.55
2	1990–05–01 12:00:00.000	15.73
...		

The definition of Π in [29, 30] is used. The inclusion of ε_r and ql_r in Definition 7 means that different segments may be retrieved for the same E due to different metric requirements. For example, if a query's metric constraints specify that no error is allowed and more recent data points are required, it is necessary to retrieve additional segments. However, prediction can be used to significantly reduce the latency if the same query is issued with metric constraints that allow the query result to have some error. The arrival process ξ is a general process that determines when the query is executed. Two special cases exist in which queries are deterministic: one-time queries that are only executed once have probability 1 at time t and 0 everywhere else, while periodic queries follow a deterministic process with period τ, such that the probability of the query being executed at time t is 1 if $t = k\tau, \forall k \in \mathbb{Z}$, and 0 otherwise. In the most general case, the arrival process can represent the decisions of the User, which can change over time or even aggregate multiple Users with the same query. For query processing, each level of tiered storage is represented as individual Parts with different latency. As most use cases include multiple different queries a *query workload* is defined as:

Definition 8
Query Workload: A *query workload* is a set of queries $W = \{Q_1, \ldots, Q_n\}$ where $|W| \geq 1$.

The queries in a query workload is evaluated as shown Algorithm 1 with the required segments retrieved from each Part as shown in Algorithm 2. However, the latency and error of the query results depend entirely on which segments are available on each Part while the query is being evaluated, and if the query's metric constraints allow prediction to be used. Thus, to optimize the evaluation of a query workload, data acquisition must be optimized according to that query workload. The specific optimizations that can be performed depend on the amount of knowledge that is available about the query workload, e.g., by knowing when and what queries will be executed an optimized sensor data acquisition scheme can be derived as described in Section 5.2.

Algorithm 1 Execution of a query Q over TS_R. Q is repeated according to ξ and E is evaluated using commonly known methods

Input: The query's relational expression E
The query's maximum error ε_r
The query's maximum latency ql_r
Output: A result set RS if ε_r and ql_r can be satisfied, if not an *error*

1: **Function** *execute_query_with_constraints*(E, ε_r, ql_r) **is**
2: return_error_after_time_interval(ql_r) ▷ Asynchronous
3: $f_p \leftarrow$ previous_part()
4: $f_n \leftarrow$ current_part()
5: $P \leftarrow$ extract_predicates(E)
6: $n\_ql_r \leftarrow ql_r -$ execution_time()
7: $RS \leftarrow$ read_or_predict(f_p, f_n, P, ε_r, n_ql_r) **on** f_n
8: **return** evaluate(E, RS)

Algorithm 2 Retrieval of the segments required to evaluate Q

Input: The previous Part f_p
The next Part f_n
The query's predicates P
The query's maximum error ε_r
The query's maximum latency ql_r
Output: A result set RS if ε_r and ql_r can be satisfied, if not an *error*

1: **Function** *read_or_predict*(f_p, f_n, P, ε_r, ql_r) $\rightarrow RS$ **is**
2: return_error_after_time_interval(ql_r) ▷ Asynchronous
3: **if** no_relevant_data_points(P) **then**
4: $RS_l \leftarrow \{\}$
5: **else if** predictable(P, ε_r) **then**
6: $RS_l \leftarrow$ predict(P, ε_r)
7: **else if** stored(P, ε_r) **then**
8: $RS_l \leftarrow$ read(P, ε_r) ▷ Sensors may be inaccurate
9: **else if** unavailable(f_p) **then**
10: **return** error() **to** f_n
11: $n\_f_p \leftarrow$ previous_part()
12: $n\_f_n \leftarrow$ current_part()
13: $n\_ql_r \leftarrow ql_r -$ execution_time()
14: $RS_p \leftarrow$ read_or_predict(n_f_p, n_f_n, P, ε_r, n_ql_r) **on** f_p
15: **return** $RS_l \cup RS_p$ **to** f_n

Table E.2: The parameters required to instantiate SENDAI

Name	Description
l_s	Time required by the Sensor to sample the State.
ε_s	Error added by the Sensor when sampling the State.
t_{ss}–t_{se}	The time interval for which the Sensor can provide segment.
l_{sp}	Time required by the Sensor to a predict a segment.
ε_{sp}	Error added by the Sensor when predicting a segment.
l_n	Time required by the edge to retrieve a segment.
ε_n	Error added by the Edge Node when retrieving a segment.
b_n	The bandwidth for the link between the Sensor and the Edge Node.
t_{ns}–t_{ne}	The time interval for which the Edge Node can provide segment.
l_{np}	Time required by the edge to predict a segment.
ε_{np}	Error added by the Edge Node when predicting a segment.
l_d	Time required by the Cloud Node to retrieve a segment.
ε_d	Error added by the Cloud Node when retrieving a segment.
b_d	The bandwidth for the link between the Edge Node and the Cloud Node.
t_{ds}–t_{de}	The time interval for which the Cloud Node can provide segment.
l_{dp}	Time required by the Cloud Node to predict a segment.
ε_{dp}	Error added by the Cloud Node when predicting a segment.
l_q	Time required by the Cloud Node to receive and respond to query.
ε_q	Error added by the Cloud Node when answering a query.
b_q	The bandwidth for the link between the Cloud Node and the User.

4.4 Summary

In summary, SENDAI describes sensor data acquisition and analytics as a sequence of Parts with sampling latency and error, prediction latency and error, the time interval each Part can provide segments for, and the bandwidth between each pair of subsequent Parts. Queries with latency and error constraints can be evaluated using the information in SENDAI and a recursive algorithm for retrieving the required segments. The set of parameters required to instantiate SENDAI for a specific use case is shown in Table E.2.

5 Sensor Data Acquisition Scheme

5.1 Periodic Sensor Data Acquisition

As a baseline for sensor data acquisition, it is assumed that the segments are collected as often as possible using periodic sampling and a predetermined schedule for trans-

missions, which does not take into account any semantic considerations, i.e., the query information or the values of the segments when sampling. This baseline was chosen as it matches the sensor data acquisition method used in the two use cases. Ideally, all of the Sensors can be sampled at a very high frequency and all of the segments can be transmitted to the Cloud Nodes using periodic sampling and a scheduled transmission scheme. However, in practice, there is always a bottleneck that limits the number of segments that can be collected. This bottleneck is often the communication link between two Parts. When a communication bottleneck prevents all of the segments from being transmitted, it is equivalent to having a maximum communication budget. So in the periodic sampling scheme, the full communication budget is used to sample all of the Sensors at a constant interval. Since the sampling is *periodic*, it will be *push-based* in most cases. However, *pull-based* sampling can make the scheduling of wireless Sensors easier if the internal clocks in the Sensors are very inaccurate and requests for new segments are very cheap. For this case, it is better to use periodic pull-based sampling to avoid packet collisions.

5.2 Semantic Sensor Data Acquisition

The sampling scheme used can be optimized based on information about which collected segments will be used for which query. Specifically, an efficient sampling scheme can be derived if all of the queries that will be executed are known beforehand. The method for deriving a sampling scheme from a query workload is based on our previous work. In [12], it is shown that knowing when queries will request specific segments can be used to reduce the latency of queries in multiple different scenarios. For example, scenarios with unreliable communication links and scenarios where only a statistical model is known for the query arrival time and not the specific times that the queries will arrive at. In [15], a method is presented for choosing which additional segments to pull to minimize the error for several different aggregate queries. These methods can be combined with other methods, e.g., the ones described in [6]. In addition, if information about the actual values is available, graph sampling methods as presented in [31, 32] can be used. However, these require that a suitable graph is known or can be constructed for a use case. There are three different scenarios to consider for semantic sampling based on the knowledge that SENDAI has about the query processes.

Full Knowledge Semantic Acquisition If the ξ **processes provides completely deterministic information about future query arrival times**, it is possible to derive a *push-based* sampling scheme that minimizes the number of queries that cannot be answered for an instance of SENDAI with a given communication budget and an assignment of latencies and errors to each Part. The fixed schedule for the sampling avoids interference and minimizes the bandwidth used and the error of the query results. As for the periodic scheme, a *pull-based* scheme could perform even better, by starting

from the Sensor with the highest uncertainty and deciding which Sensor to request data from at every transmission opportunity in an adaptive way that considers the actual value of the State monitored by the Sensors that have already replied. This can be further optimized using a tree search algorithm that optimizes the decision path with a fixed horizon corresponding to the instant of the next query [2].

Blind Semantic Acquisition If the ξ **processes provides absolutely no information about future query arrival times**, optimizing sampling for a particular query becomes highly suboptimal. For example, if the sampling is optimized for a particular query and that query does not arrive when expected or a different query arrives unexpectedly while all of the bandwidth of the communication links is being used. In this scenario there might not be enough bandwidth left to pull the segments required to answer the received query. Thus, it is best to allocate all of the communication budget to *pull-based* sampling, so the required segments can be collected efficiently whenever a query arrives at the Cloud Nodes. In this case, AoI optimization might work best, or a scheme aimed at minimizing the error on the system state, as described in our previous work [15]. This case is the opposite extreme to full knowledge acquisition, and while the other represents an upper bound to the achievable performance of semantic sampling, this is the lower bound, as the sampling is better than the baseline periodic scheme, but can only exploit information about the underlying process measured by the Sensors, and has no knowledge of the query process.

Partial Knowledge Semantic Acquisition In-between these two extremes are the cases where some queries have deterministic arrival times, some queries have arrival times with a high level of certainty, and some queries have arrival times with a high level of uncertainty. In these cases, the optimal solution is to allocate some of the communication budget to a *push-based* sampling scheme focused on replying to as many of the queries with deterministic arrival times and arrival times with high certainty as possible, and a part of the budget for *pulling additional segments* for the queries with uncertain arrival times. This is the most realistic scenario, but also the most complex, and will require some form of learning-based optimization to achieve the best possible performance. Below are two examples that are based on the wind turbine use case from Section 3.1 and the smart meter use cases from Section 3.2 and shows how semantic sampling can improve sensor data acquisition using information about when and which queries will be executed.

5.3 Optimizing the Wind Turbine Use Case

The first example is based on the wind turbine use case from Section 3.1. In this example, two types of queries are executed: *ad-hoc* queries requesting the data point with the *maximum value* among all of the Sensors, and *periodic queries* requesting the *latest*

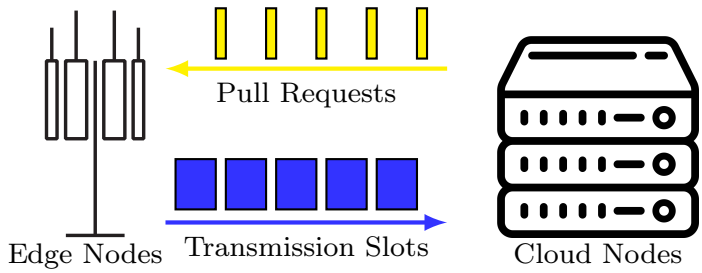


Fig. E.6: Transmission scheme that allows for preparation for predictable queries and pull requests for unpredictable queries.

data point for each of the Sensors. In the wind turbine use case, the communication bottleneck is the communication link between the Edge Nodes and the Cloud Nodes. So the Edge Nodes can collect data points from all of the Sensors at a high frequency but can only transmit a fraction of those data points to the Cloud Nodes. We consider a communication scheme where the Edge Nodes listen for a pull request for a short time followed by a longer transmission slot as shown in Figure E.6. In addition, it is assumed that transmitting the data point with the maximum value among all of the Sensors requires one transmission slot and that transmitting the latest data point for each of the Sensors requires several transmission slots. To efficiently answer the queries requesting the latest data point for each of the Sensors, it is necessary to prepare for the queries by starting to transmit the required data points several transmission slots before each query arrives. However, it is not possible to answer any pull requests in the transmission slots allocated to this preparation, except by using prediction. Thus, there is a trade-off between how early to start preparing for the queries requesting the latest data points for each of the Sensors and how many opportunities there are for pulling data for the queries for the data point with the maximum value. Starting the preparation early decreases the accuracy of the first query type because the AoI is higher, while not leaving enough pull opportunities increases the latency for the second query type, because the reply has to wait. If the preparation for the predictable queries is started early enough to leave opportunities to answer pull requests, the latency of the queries requesting the data points with the maximum value is kept low at the expense of the accuracy for the queries requesting the latest data point for each of the Sensors, and there is also a risk that there will be an empty transmission slot right before a query. If too few slots are reserved for answering the pull requests, the accuracy will be high for the queries requesting the latest data points for each of the Sensors, but the latency will be high for the queries requesting the data point with the maximum value as it is necessary to wait until the scheduled transmissions are complete before it is possible to reply. This use case corresponds to a partial knowledge scenario, so the actual semantic

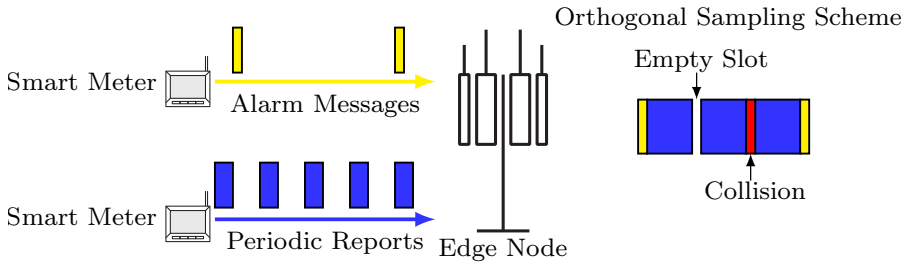


Fig. E.7: Sampling scheme that accommodates periodic reporting and occasional alarm messages.

sampling scheme is complex, but the periodicity of predictable queries can be exploited as in Figure E.6.

5.4 Optimizing the Smart Meter Use Case

The second example is based on the smart meter use case from Section 3.2. In this use case, *periodic* reporting is combined with *unpredictable alarms* that need to be delivered with relatively *low latency*. For this scenario, reserving some transmission slots for alarms can be an efficient solution as it allows periodic reporting to be scheduled in advance. Thus, packet collisions are rare and AoI is kept low, at the cost of keeping a certain fraction of the transmission slots unused. If a smart meter raises an alarm, it can try to transmit it in one of these unused transmission slots. As these transmission slots are never used for periodic reporting, an alarm packet can only collide with other alarm packets. Thus, there is a trade-off between the freshness of the data points produced by the smart meter for periodic reporting and the latency and risk of packet collisions for the alarm packet. If a high number of transmission slots are unused, both the latency and the risk of collision for alarm packets are reduced. However, a high number of unused transmission slots also increases the AoI for the data points collected during periodic reporting, as they have fewer resources, thus decreasing the accuracy or increasing the latency for queries. On the other hand, reserving too many transmission slots for alarm messages causes inefficient use of bandwidth (represented by the empty transmission slot) while reserving too few can cause collisions (represented by the red transmission slot) which increases the latency of the alarm packets and reduces the probability that they will be delivered. The described sampling scheme is shown in Figure E.7. An analysis for this type of sampling scheme is presented in our previous work [33].

6 Related Work

6.1 Sensor Data Acquisition

The acquisition of data from sensors needs to be optimized for several metrics, which include both the types and frequencies of the queries to the edges node and considerations specific to the sensors themselves, such as the limited transmission bandwidth and the energy consumption requirements.

Traub et al. [6] proposed using adaptive user-defined sampling functions to sample sensors based on the demand of concurrently running queries. Thus, through adaptive sampling, local filtering, and sharing of sensor data between queries the amount of collected sensor data is reduced. Hülsmann et al. [7] demonstrated the method, while Hülsmann et al. [8] extended the method to automatically adjust the allowed sample time for a sensor based on the values it produces. Gil et al. [34] proposed only transmitting and storing significant data points by using multiple data point selection algorithms and then selecting the best algorithm for each time interval.

The concept of AoI [35] from the communications literature is also often used in context of data acquisition: by optimizing the AoI, the system can improve the freshness of the data from each sensor [24], reducing the error on the queries. Our previous works [12, 36] combine the AoI and the awareness of the query process to optimize a metric called Query AoI (QAoI), which represents the freshness of the data at the instant a query comes. Other sensor selection methods can use statistics about the sensors [34] or an estimate of the state of the underlying process [15] to select which sensor to poll in a pull-based system, and the general division we mentioned in Section 4 generally holds throughout the communication literature.

6.2 Sensor Data Analytics

The development of Database Management Systems (DBMSs) optimized for time series have received signification focus due to the increasing amount of sensor data being produced. For a in-depth survey of these TSMSs see [5].

Most TSMSs are only optimized for deployment on the edge nodes or the cloud nodes. However, a few TSMSs support sensor data acquisition, transmission, and management. Respawn [37] is designed to be deployed on both edge nodes and cloud nodes. Data points are continuously transmitted from the edge nodes to the cloud nodes and a Dispatcher routes query to the correct nodes. Storacle [38] is designed to be deployed on edge nodes to monitor smart grids. Data points are transmitted to cloud nodes for permanent storage and offline processing. Data points are not immediately deleted when transmitted to support efficiently executing queries on the edge nodes. Apache IoTDB [39] is designed to support three deployment models: as an embedded TSMS on low-powered edge nodes, as a single node TSMS on high-powered edge nodes, and as

a distributed TSMS on cloud nodes. A File Sync module transmits data between the deployed instances. VergeDB [40] does not support transmitting Data points between instances, it is designed to optimize the representation used for data points by the edge nodes based on the requirements of the analytics performed in the cloud. Also, while not a TSMS the GLEAN [41] framework compresses data points using Generalized Deduplication [42, 43] and stores the resulting Bases on the edge nodes and the Deviations in the cloud. GLEAN also supports clustering using the compressed data [41, 44].

The use of prediction and approximation generally reduces latency and the of amount storage required. TimeTravel [45] is designed to provide a uniform interface for exact queries on historical data points, prediction of historical data points, and forecasting. The prediction and forecasting are performed using models build from the time series, e.g., ARIMA, and organized in a novel Hierarchical Model Index. Like TimeTravel, tspDB [21] supports both exact queries on historical data points, prediction of historical data points, and forecasting. However, instead of using models like ARIMA for prediction and forecasting, it uses a novel incremental method based on matrix factorization. RoundRobinson [46] is an implementation of a formalism for TSMSs that enable multiresolution storage of time series. For example, recent data may be stored at high-resolution with the resolution decreasing as the data age. Tristan [47] is designed to efficiently store and analyze time series using dictionary compression. The compressed representation is computed using Matching Pursuit [48]. The CORAD [49] compression method extends Tristan's compression method to exploit that time series often are correlated. ModelarDB [25, 50, 51] stores time series using different types of models, e.g., constant and linear functions. Multiple model types are used for each time series as they change over time. Similar time series can also be grouped and compressed together to further reduce the amount of storage required.

7 Conclusion and Future Work

Despite the ever-increasing number of deployed sensors and the huge amounts of data points they produce, most methods for sensor data acquisition and analytics focus only on either acquisition or analytics instead of taking a holistic approach, thus limiting the optimizations that can be performed. Motivated by this problem, this paper proposed the Framework for joint Sensory Data Acquisition and Analysis (SENDAI), an integrated framework for reasoning about sensor data acquisition and analytics together. SENDAI was designed based on an analysis of two very different use cases using the information provided by multiple industrial partners, specifically monitoring wind turbines and utility consumption. It was shown how SENDAI can be used to optimize sensor data acquisition to efficiently respond to a query workload. This demonstrates that SENDAI can be used to reason about sensor data acquisition and analytics together,

In future work, we plan to extend SENDAI in multiple directions: (i) increase the detail of SENDAI by adding more Parts; (ii) take more resource limitations into account,

e.g., the amount of energy and storage available to each Part; (iii) allow SENDAI to be instantiated with different error functions for each Part; (iv) allow the error and latency for each Part to be defined as a distribution instead of a static value; (v) support selecting an efficient sampling scheme for additional types of queries.

Acknowledgements

This work was supported by the Danish Council for Independent Research under grant number 8022-00284B (SEMIOTIC), by the European Commission as part of the Horizon 2020 program under grant number 957218 (IntellIoT) and grant number 957345 (MORE), and by the Velux Foundation under Villum Investigator grant WATER. We also thank our industrial partners for the information provided about the two use cases.

References

[1] A. B. Sharma, F. Ivančić, A. Niculescu-Mizil, H. Chen, and G. Jiang, "Modeling and Analytics for Cyber-Physical Systems in the Age of Big Data," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 4, pp. 74–77, 2014.

[2] P. Popovski, F. Chiariotti, K. Huang, A. E. Kalør, M. Kountouris, N. Pappas, and B. Soret, "A Perspective on Time towards Wireless 6G," *CoRR*, vol. abs/2106.04314, 2022.

[3] P. Esling and C. Agon, "Time-Series Data Mining," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–34, 2012.

[4] A. Fakhrazari and H. Vakilzadian, "A Survey on Time Series Data Mining," in *2017 IEEE International Conference on Electro Information Technology*, 2017, pp. 476–481.

[5] S. K. Jensen, T. B. Pedersen, and C. Thomsen, "Time Series Management Systems: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2581–2600, 2017.

[6] J. Traub, S. Breß, T. Rabl, A. Katsifodimos, and V. Markl, "Optimized On-Demand Data Streaming from Sensor Nodes," in *Proceedings of the 2017 Symposium on Cloud Computing*. ACM, 2017, pp. 586–597.

[7] J. Hülsmann, J. Traub, and V. Markl, "Demand-based Sensor Data Gathering with Multi-Query Optimization," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2801–2804, 2020.

[8] J. Hülsmann, C. Li, J. Traub, and V. Markl, "Automatic Tuning of Read-Time Tolerances for Optimized On-Demand Data-Streaming from Sensor Nodes," in *Proceedings of the 24th International Conference on Extending Database Technology*. OpenProceedings.org, 2021, pp. 517–522.

[9] 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, no. 4, pp. 1897–1910, 2016.

[10] B. Zhou and W. Saad, "Joint Status Sampling and Updating for Minimizing Age of Information in the Internet of Things," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 7468–7482, 2019.

[11] C. Li, S. Li, Y. Chen, Y. T. Hou, and W. Lou, "Minimizing Age of Information Under General Models for IoT Data Collection," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2256–2270, 2019.

[12] F. Chiariotti, J. Holm, A. E. Kalør, B. Soret, S. K. Jensen, T. B. Pedersen, and P. Popovski, "Query Age of Information: Freshness in Pull-Based Communication," *IEEE Transactions on Communications*, vol. 70, no. 3, pp. 1606–1622, 2022.

[13] F. Z. Djiroun and D. Djenouri, "MAC Protocols With Wake-Up Radio for Wireless Sensor Networks: A Review," *IEEE Communications surveys & tutorials*, vol. 19, no. 1, pp. 587–618, 2016.

[14] P. R. Jhunjhunwala, B. Sombabu, and S. Moharir, "Optimal aoi-aware scheduling and cycles in graphs," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1593–1603, 2019.

[15] F. Chiariotti, A. E. Kalør, J. Holm, B. Soret, and P. Popovski, "Scheduling of sensor transmissions based on Value of Information for summary statistics," *IEEE Netw. Letters*, vol. 4, no. 2, pp. 92–96, May 2022.

[16] A. Munari, "Modern Random Access: An Age of Information Perspective on Irregular Repetition Slotted ALOHA," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3572–3585, 2021.

[17] X. Chen, K. Gatsis, H. Hassani, and S. S. Bidokhti, "Age of Information in Random Access Channels," in *International Symposium on Information Theory*. IEEE, 2020, pp. 1770–1775.

[18] S. Sathe, T. G. Papaioannou, H. Jeung, and K. Aberer, "A Survey of Model-based Sensor Data Acquisition and Management," in *Managing and Mining Sensor Data*. Springer, 2013, pp. 9–50.

[19] T. Palpanas, "Real-Time Data Analytics in Sensor Networks," in *Managing and Mining Sensor Data*. Springer, 2013, pp. 173–210.

[20] C. C. Aggarwal, "Mining Time Series Data," in *Data Mining: The Textbook*. Springer, 2015, pp. 457–491.

[21] A. Agarwal, A. Alomar, and D. Shah, "tspDB: Time Series Predict DB," in *NeurIPS 2020 Competition and Demonstration Track*. PMLR, 2020, pp. 27–56.

[22] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: NEURAL BASIS EXPANSION ANALYSIS FOR INTERPRETABLE TIME SERIES FORECASTING," in *8th International Conference on Learning Representations*. OpenReview.net, 2020, pp. 1–21.

[23] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, 2021, pp. 11 106–11 115.

[24] R. Devassy, G. Durisi, G. C. Ferrante, O. Simeone, and E. Uysal, "Reliable transmission of short packets through queues and noisy channels under latency and peak-age violation guarantees," *IEEE Journ. on Selected Areas in Communications*, vol. 37, no. 4, pp. 721–734, Feb. 2019.

[25] S. K. Jensen, T. B. Pedersen, and C. Thomsen, "Scalable Model-Based Management of Correlated Dimensional Time Series in ModelarDB$_+$," in *37th IEEE International Conference on Data Engineering*. IEEE, 2021, pp. 1380–1391.

[26] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, p. 377–387, 1970.

[27] ——, "Further Normalization of the Data Base Relational Model," *Research Report / RJ / IBM*, vol. RJ909, 1971.

[28] ——, "Relational Completeness of Data Base Sublanguages," *Research Report / RJ / IBM*, vol. RJ987, 1972.

[29] V. Harinarayan and A. Gupta, "Generalized Projections: a Powerful Query-Optimization Technique," Stanford University, Department of Computer Science, Tech. Rep. STAN-CS-TN-94-14, 1994.

[30] A. Gupta, V. Harinarayan, and D. Quass, "Aggregate-Query Processing in Data Warehousing Environments," in *Proceedings of 21th International Conference on Very Large Data Bases*. Morgan Kaufmann, 1995, pp. 358–369.

[31] J. Holm, F. Chiariotti, M. Nielsen, and P. Popovski, "Lifetime maximization of an internet of things (iot) network based on graph signal processing," *IEEE Communications Letters*, 2021.

[32] A. Chiumento, N. Marchetti, and I. Macaluso, "Energy efficient wsn: a cross-layer graph signal processing solution to information redundancy," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2019, pp. 645–650.

[33] F. Chiariotti, I. Leyva-Mayorga, Č. Stefanović, A. E. Kalør, and P. Popovski, "Spectrum Slicing for Multiple Access Channels with Heterogeneous Services," *Entropy*, vol. 23, no. 6, p. 686, 2021.

[34] A. Gil, M. Quartulli, I. G. Olaizola, and B. Sierra, "Towards Smart Data Selection From Time Series Using Statistical Methods," *IEEE Access*, vol. 9, pp. 44 390–44 401, 2021.

[35] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[36] J. Holm, A. E. Kalør, F. Chiariotti, B. Soret, S. K. Jensen, T. B. Pedersen, and P. Popovski, "Freshness on Demand: Optimizing Age of Information for the Query Process," in *IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[37] M. Buevich, A. Wright, R. Sargent, and A. Rowe, "Respawn: A Distributed Multi-Resolution Time-Series Datastore," in *Proceedings of the IEEE 34th Real-Time Systems Symposium*. IEEE, 2013, pp. 288–297.

[38] S. Cejka, R. Mosshammer, and A. Einfalt, "Java embedded storage for time series and meta data in Smart Grids," in *2015 IEEE International Conference on Smart Grid Communications*. IEEE, 2015, pp. 434–439.

[39] C. Wang, X. Huang, J. Qiao, T. Jiang, L. Rui, J. Zhang, R. Kang, J. Feinauer, K. A. McGrail, P. Wang, D. Luo, J. Yuan, J. Wang, and J. Sun, "Apache IoTDB: Time-series Database for Internet of Things," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2901–2904, 2020.

[40] J. Paparrizos, C. Liu, B. Barbarioli, J. Hwang, I. Edian, A. J. Elmore, M. J. Franklin, and S. Krishnan, "VergeDB: A Database for IoT Analytics on Edge Devices," in *11th Conference on Innovative Data Systems Research*, 2021, pp. 1–8.

[41] A. Hurst, D. E. Lucani, I. Assent, and Q. Zhang, "GLEAN: Generalized Deduplication Enabled Approximate Edge Analytics," *IEEE Internet of Things Journal*, pp. 1–15, 2022.

[42] R. Vestergaard, Q. Zhang, and D. E. Lucani, "Generalized Deduplication: Bounds, Convergence, and Asymptotic Properties," in *IEEE Global Communications Conference*. IEEE, 2019, pp. 1–6.

[43] ——, "Lossless Compression of Time Series Data with Generalized Deduplication," in *IEEE Global Communications Conference*. IEEE, 2019, pp. 1–6.

[44] A. Hurst, Q. Zhang, D. E. Lucani, and I. Assent, "Direct Analytics of Generalized Deduplication Compressed IoT Data," in *IEEE Global Communications Conference*. IEEE, 2021, pp. 1–6.

[45] M. E. Khalefa, U. Fischer, T. B. Pedersen, and W. Lehner, "Model-based Integration of Past & Future in TimeTravel," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1974–1977, 2012.

[46] A. L. Serra, S. Vila-Marta, and T. Escobet Canal, "Formalism for a multiresolution time series database model," *Information Systems*, vol. 56, pp. 19–35, 2016.

[47] A. Marascu, P. Pompey, E. Bouillet, M. Wurst, O. Verscheure, M. Grund, and P. Cudre-Mauroux, "TRISTAN: Real-Time Analytics on Massive Time Series Using Sparse Dictionary Compression," in *Proc. 2014 IEEE Int. Conf. on Big Data*. IEEE, 2014, pp. 291–300.

[48] S. Mallat and Z. Zhang, "Matching Pursuits With Time-Frequency Dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[49] A. Khelifati, M. Khayati, and P. Cudré-Mauroux, "CORAD: Correlation-Aware Compression of Massive Time Series using Sparse Dictionary Coding," in *2019 IEEE International Conference on Big Data*. IEEE, 2019, pp. 2289–2298.

[50] S. K. Jensen, T. B. Pedersen, and C. Thomsen, "ModelarDB: Modular Model-Based Time Series Management with Spark and Cassandra," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1688–1701, 2018.

[51] ——, "Demonstration of ModelarDB: Model-Based Management of Dimensional Time Series," in *Proceedings of the 2019 International Conference on Management of Data*. ACM, 2019, pp. 1933–1936.

Paper F

Scheduling of Sensor Transmissions Based on Value of Information for Summary Statistics

Federico Chiariotti, Anders E. Kalør, Josefine Holm, Beatriz Soret and Petar Popovski

The paper has been published in the
IEEE Networking Letters, vol. 4, no. 2, pp. 92–96, 2022.

The layout has been revised.

Abstract

The optimization of VoI in sensor networks integrates awareness of the measured process in the communication system. However, most existing scheduling algorithms do not consider the specific needs of monitoring applications, but define VoI as a generic Mean Square Error (MSE) of the whole system state regardless of the relevance of individual components. In this work, we consider different summary statistics, i.e., different functions of the state, which can represent the useful information for a monitoring process, particularly in safety and industrial applications. We propose policies that minimize the estimation error for different summary statistics, showing significant gains by simulation.

1 Introduction

Over the past few years, the unprecedented development of the IoT has made the remote estimation of stochastic processes a central problem in communications and automation [1], where a set of sensors transmit observations to a central Base Station (BS). The possibility to process sensor data either at the BS or in a distributed fashion through in-network processing [2] has led the research community to focus extensively on the scheduling of sensor updates in severely resource-constrained wireless network environments.

For a wide range of remote estimation problems, the *freshness* of the observations at the BS is a good proxy for the estimation quality. This promotes AoI [3] as a measure of the time that has passed since the last update from a given sensor. However, if the destination has a model of the observed processes, it is often better to directly minimize the uncertainty of the process estimates instead of the AoI [4]. The problem of scheduling IoT sensors with this goal has been considered for several different policies [5, 6], whose objective is to minimize the MSE of a Kalman filter, considering communication constraints. More recently, the problem of minimizing the MSE of the process estimates has been referred to as VoI [4]. A recent work [7] tries to maximize the accuracy of a more complex unscented filter, aiming at optimal sensor selection for maneuvering tasks, and VoI can also be used for data muling applications in underwater or drone networks [8]. Another interesting twist to this is the application of VoI concepts not over time, but in space, placing sensors in the positions that will result in the highest overall accuracy for the estimation of a spatial process [9].

However, there are cases where minimizing the MSE is not be the best thing to do: for example, if the application needs to compute a non-linear function of the state, such as the maximum value among all sensors. While minimizing the MSE implicitly gives equal value to all sensors, some might have a larger weight in the non-linear function (e.g., sensors with a higher value for the maximum function). Examples in industrial settings

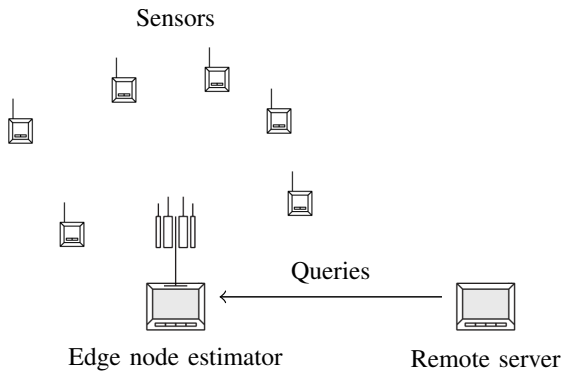


Fig. F.1: Representation of the scenario.

include: (1) triggering a safety warning if the temperature of any of the components in a machine reaches a safety limit, (2) monitoring if the difference in the strain on different parts of a structure is outside the design parameters. Such a scenario is represented in Fig. F.1: the remote server sends queries to the BS, which correspond to the non-linear function, and the BS needs to schedule transmission so as to maximize the accuracy. This setup was also used in our previous work [10]. The scheduling in this scenario is driven by the BS, which selects the sensor that it believes to have the most useful information at each time slot; the opposite scenario, in which sensors themselves decide whether to transmit or not, is an interesting but different problem, as it requires sensors to maintain an estimate of the system state and a decision algorithm, which consume energy, as well as to coordinate among themselves to avoid collisions. Our scenario is directly applicable to wake-up radio [11, 12] and similar schemes with low-power sensors.

We propose heuristic strategies to schedule sensor updates in a linear dynamic system, which explicitly aim to minimize the error of various summary statistics. We derive the one-step optimal strategies for some well-known function, and give a general Monte Carlo-based algorithm that can deal with different query functions. The simulations show that the proposed strategies can significantly reduce the error on a number of summary statistics, with more significant gains in case of highly non-linear summary statistics.

The rest of this letter is organized as follows. The system model is presented in Sec. 2, and one-step policies for various summary statistics are derived in Sec. 3. Numerical results are presented in Sec. 4, and finally Sec. 5 concludes the paper and presents some possible avenues of future work.

2 System Model

We consider a system with N sensors, which are connected through time-slotted wireless links to a BS equipped with computing and storage resources. Without loss of generality, we assume that the time slots occur at $t = 1, 2, \ldots$ and the sensors are indexed by $n = 1, \ldots, N$. We assume that each sensor observes a value in an N-dimensional process, whose state $\mathbf{x}(t) = [x_1, \ldots, x_N]^T$ evolves according to

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t-1) + \mathbf{v}(t), \tag{F.1}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the transition matrix, $\mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_v})$ is the process noise with covariance matrix $\mathbf{\Sigma_v} \in \mathbb{R}^{N \times N}$, and $\mathbf{x}(0) = \mathbf{0}$. The sensors observe the processes with additive white Gaussian measurement noise $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_w})$, i.e., $\mathbf{y}(t) = \mathbf{x}(t) + \mathbf{w}(t)$. In general, the covariance matrices $\mathbf{\Sigma_v}$ and $\mathbf{\Sigma_w}$ are not diagonal. Note that although we assume that the number of sensors is equal to the dimension of the process (to simplify the notation), the analysis can be easily extended to more general observable systems.

We consider a Time Division Multiple Access (TDMA) air interface, in which each time slot, t, contains a downlink phase and an uplink phase. The downlink is used by the BS to schedule the sensor, $a(t)$, that transmits its observation $y_{a(t)}(t)$ in the uplink phase. The channel is modeled as a packet erasure channel with error probability $\varepsilon_{a(t)}$, which captures errors both in the transmission of the scheduling decision and the observation. We also assume that the process dynamics are known to the BS, a standard assumption in Kalman filtering, which is practical if the monitored system is well-understood, even if its instantaneous state is hard to measure directly. This condition is common for many IoT applications [13, 14], in which well-known processes are estimated by sensors over wide areas. We also denote the row vector of length N whose only non-zero value is the n-th, which is 1, as $\mathbf{1}_n$, and the $N \times N$ identity matrix as $\mathbf{I}_N$.

2.1 Kalman Filter Estimation

We assume that the BS maintains a distribution over its belief of the state $p(\mathbf{x}(t))$ using a Kalman filter. The Kalman filter is the Minimum Mean Square Error (MMSE) estimator for the model defined in (F.1) [15], in which case $p(\mathbf{x}(t)) \sim \mathcal{N}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t))$. The mean vector $\hat{\mathbf{x}}(t)$ and the covariance matrix $\boldsymbol{\psi}(t)$ are updated at each timestep t based on the outcome of the scheduling process. The Kalman filter operates in two steps: a *prior* update, which only depends on the system statistics, and a *posterior* update, which integrates new observations. The prior update operation is given by:

$$\hat{\mathbf{x}}(t) = \mathbf{A}\hat{\mathbf{x}}(t-1) \triangleq \hat{\mathbf{x}}_{\mathrm{F}}(t), \tag{F.2}$$

$$\boldsymbol{\psi}(t) = \mathbf{A}\boldsymbol{\psi}(t-1)\mathbf{A}^T + \mathbf{\Sigma_v} \triangleq \boldsymbol{\psi}_{\mathrm{F}}(t). \tag{F.3}$$

If the transmission of the update fails, an event we denote as F, the BS can only rely on the prior update for its estimate. If the update is received, it can be used to improve the estimate. We then compute the Kalman filter gain $\mathbf{k}(t)$:

$$\mathbf{k}(t) = \boldsymbol{\psi}_{\mathrm{F}}(t)\mathbf{1}_{a(t)}^{T}\left[\mathbf{1}_{a(t)}\left(\boldsymbol{\psi}_{\mathrm{F}}(t)+\boldsymbol{\Sigma}_{\mathbf{w}}\right)\mathbf{1}_{a(t)}^{T}\right]^{-1}. \tag{F.4}$$

We finally get the updated estimate in case of a success event:

$$\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}_{F}(t) + \mathbf{k}(t)\mathbf{1}_{a(t)}(\mathbf{y}(t) - \hat{\mathbf{x}}_{F}(t)) \triangleq \hat{\mathbf{x}}_{\mathrm{S},a(t)}(t) \tag{F.5}$$

$$\boldsymbol{\psi}(t) = (\mathbf{I}_{N} - \mathbf{k}(t)\mathbf{1}_{a(t)})\boldsymbol{\psi}_{\mathrm{F}}(t) \triangleq \boldsymbol{\psi}_{\mathrm{S},a(t)}(t), \tag{F.6}$$

Note that the recursive structure of the Kalman filter and the independence of the transmission errors imply that $\hat{\mathbf{x}}(t)$ and $\boldsymbol{\psi}(t)$ are sufficient statistics for the state estimate given the full history $\mathcal{H}(t)$ of past actions and observations.

2.2 Summary Statistics

Unlike the majority of VoI applications, in which the BS aims to minimize the MSE of $\hat{\mathbf{x}}(t)$, we consider the case in which an external user requests *summary statistics* about the state of the system: these correspond to a predefined, fixed function of the system state, e.g., the average value or the number of states with values within a given interval. Formally, we define a summary statistic as a function $z : \mathbb{R}^{N} \to \mathbb{R}$ of the true state $\mathbf{x}(t)$. However, because $\mathbf{x}(t)$ is unknown to the BS, it can only provide an *approximate* answer to the query based on its state belief $p(\mathbf{x}(t))$. We will consider estimators of the summary statistics on the form

$$\hat{z}(t) = \mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\hat{\mathbf{x}}(t),\boldsymbol{\psi}(t))}[z(\mathbf{x})], \tag{F.7}$$

which corresponds to the minimum MSE estimator of $z(\mathbf{x}(t))$, given the current observation [16, 17]. This is different from minimizing the MSE of $\hat{\mathbf{x}}(t)$, particularly when function $z(\cdot)$ is non-linear or the sensors have different weights. We denote the squared error as $\nu_{z}(t)$:

$$\nu_{z}(t) = (z(\mathbf{x}(t)) - \hat{z}(t))^{2}. \tag{F.8}$$

3 Scheduling Strategies

In our scenario, we seek a scheduling strategy, that is, a function π_{z} from the current Kalman state (which is represented by vector $\hat{\mathbf{x}}(t)$ and matrix $\boldsymbol{\psi}(t)$) to an action $a(t)$, that minimizes the expected error for a given summary statistic z:

$$\underset{\pi_{z}\in\Pi}{\text{minimize}}\,\mathbb{E}\left[\nu_{z}(t) \mid \pi_{z}\right], \tag{F.9}$$

While we only consider the error in the next time step, the optimal solutions are expected to perform well with respect to the long-term error due to the linearity of the observed process (despite a non-linear summary statistic). Computing $\mathbb{E}\left[\nu_z(t)|a(t)\right]$ is not simple, but it can be expressed in terms of the two possible transmission outcomes:

$$\begin{aligned}\mathbb{E}\left[\nu_z(t) \mid a(t)\right] =&(1-\varepsilon_{a(t)})\,\mathbb{E}\left[\nu_z(t) \mid \hat{\mathbf{x}}_{\mathrm{S},a(t)}(t), \boldsymbol{\psi}_{\mathrm{S},a(t)}(t)\right] \\ &+\varepsilon_{a(t)}\,\mathbb{E}\left[\nu_z(t) \mid \hat{\mathbf{x}}_{\mathrm{F}}(t), \boldsymbol{\psi}_{\mathrm{F}}(t)\right],\end{aligned} \tag{F.10}$$

where the expectation is over the state evolution. Since $\boldsymbol{\psi}_{\mathrm{S},a(t)}(t)$ and $\boldsymbol{\psi}_{\mathrm{F}}(t)$ can be computed using (F.6) and (F.3), we can iterate over the possible actions and find the optimal scheduler, as long as we can estimate the MSE for a given observation. In the following, we derive the optimal schedulers for some well-known summary statistics, along with giving a Monte Carlo-based approximate scheduler that can deal with more complex statistics for which the MSE is hard to express in closed form. Using the result from (F.10) we can obtain the optimal scheduling decision at time t as:

$$a_z^*(t) = \underset{a(t)\in\{1,\dots,N\}}{\arg\min}\ \mathbb{E}\left[\nu_z(t) \mid a(t)\right]. \tag{F.11}$$

3.1 Baseline Scheduler

We start by defining our benchmark scheme, which aims to minimize the MSE between the true state $\mathbf{x}$ and the estimated state $\hat{\mathbf{x}}$. The query is then computed as in (F.7) based on the MMSE state estimate.

The squared state estimation error can be expressed as

$$\mathbb{E}\left[\nu_{\mathrm{MSE}}(t)\right] = \mathbb{E}\left[(\mathbf{x}(t)-\hat{\mathbf{x}}(t))^T(\mathbf{x}(t)-\hat{\mathbf{x}}(t))\right]. \tag{F.12}$$

Because $(\mathbf{x}(t)-\hat{\mathbf{x}}(t)) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\psi}(t))$, the expression above is equivalent to the trace of the covariance matrix $\boldsymbol{\psi}$ [18]:

$$\mathbb{E}\left[\nu_{\mathrm{MSE}}(t) \mid \boldsymbol{\psi}(t)\right] = \mathrm{tr}(\boldsymbol{\psi}(t)). \tag{F.13}$$

We can then use (F.11) to compute the minimum MSE schedule.

3.2 Sample Mean Scheduling

We now consider the most basic statistic, the sample mean:

$$z_{\mathrm{avg}}(\mathbf{x}(t)) = \frac{1}{N}\sum_{n=1}^{N} x_n(t). \tag{F.14}$$

The estimation error $\nu_{\mathrm{avg}}(t)$ is equal to the square of the average difference between the true and the estimated entries of $\mathbf{x}$. Since the sum of all elements in $\mathbf{x}(t)-\hat{\mathbf{x}}(t)$ is a

Gaussian random variable with zero mean and variance equal to the sum of all elements in $\boldsymbol{\psi}(t)$, we have:

$$\mathbb{E}\left[\nu_{\text{avg}}(t) \mid \boldsymbol{\psi}(t)\right] = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\psi^{(i,j)}(t)}{N^2}, \tag{F.15}$$

where $\psi^{(i,j)}(t)$ is entry (i,j) of $\boldsymbol{\psi}(t)$. We can then use the result in (F.11) to derive the one-step optimal schedule.

3.3 Sample Variance Scheduling

Another important summary statistic is the sample variance, quantifying how much the state deviates from the mean:

$$z_{\text{var}}(\mathbf{x}(t)) = \frac{1}{N-1}\sum_{n=1}^{N}\left(x_n(t) - \sum_{m=1}^{N}\frac{x_m(t)}{N}\right)^2. \tag{F.16}$$

To derive the scheduling policy, it is convenient to express $z_{\text{var}}(\mathbf{x}(t))$ in quadratic form with matrix $\mathbf{M} = \mathbf{I} - 1/N$:

$$z_{\text{var}}(\mathbf{x}(t)) = \frac{(\mathbf{M}\mathbf{x}(t))^T\mathbf{M}\mathbf{x}(t)}{N-1} = \frac{\mathbf{x}(t)^T\mathbf{M}\mathbf{x}(t)}{N-1}. \tag{F.17}$$

Taking into account the belief $p(\mathbf{x}) \sim \mathcal{N}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t))$, the expected value and variance of the sample variance are known from the literature [18]:

$$\hat{z}_{\text{var}}(t) = \frac{1}{N-1}\left(\text{tr}\left(\mathbf{M}\boldsymbol{\psi}(t)\right) + \hat{\mathbf{x}}(t)^T\mathbf{M}\hat{\mathbf{x}}(t)\right) \tag{F.18}$$

$$\mathbb{E}\left[\nu_{\text{var}}(t)|\boldsymbol{\psi}(t)\right] = \frac{2\,\text{tr}\left(\mathbf{M}\boldsymbol{\psi}(t)^2\right) + 4\hat{\mathbf{x}}(t)^T\mathbf{M}\boldsymbol{\psi}(t)\hat{\mathbf{x}}(t)}{(N-1)^2}. \tag{F.19}$$

As for the MSE and sample mean, we can now simply derive the scheduler by using this result in (F.11).

3.4 Statistic-aware Monte Carlo scheduling

We can now consider a generic summary statistic z: in the general case, computing the expected MSE can be extremely complex, or even impossible in closed form. In order to still provide an approximate scheduler, we consider Monte Carlo sampling to estimate $\mathbb{E}\left[\nu_z(t)|a(t)\right]$. This method consists of drawing M samples from the conditioned multivariate Gaussian distribution $p(\mathbf{x}|\hat{\mathbf{x}})$, with complexity $O(MN^2)$, and is guaranteed to converge to the correct estimate as $M \to \infty$ thanks to the law of large numbers [19]. This estimate can then be used in (F.11) to perform scheduling.

Algorithm 1 Monte Carlo scheduling policy

function SCHEDULE($\hat{\mathbf{x}}(t-1), \boldsymbol{\psi}(t-1), \mathbf{A}, \boldsymbol{\Sigma}_{\mathbf{v}}, \boldsymbol{\Sigma}_{\mathbf{w}}, \boldsymbol{\varepsilon}, z$)
  $\nu \leftarrow \mathbf{0}$
  for $n \in \{1, \dots, N\}$ **do**
    $\mathbf{u} \leftarrow \mathbf{0}$
    for $m \in \{1, \dots, M\}$ **do**
      $\hat{\mathbf{x}}(t-1), \boldsymbol{\psi}(t-1), \mathbf{S} \leftarrow$ PRIORUPDATE($\hat{\mathbf{x}}, \boldsymbol{\psi}, \mathbf{A}, \boldsymbol{\Sigma}_{\mathbf{v}}$)
      if RANDOM$(0,1) \geq \varepsilon_n$ **then** ▷ Update successful
        $y \leftarrow$ GAUSSIANSAMPLE($\mathbf{1}_n \hat{\mathbf{x}}, \mathbf{1}_n \mathbf{S} \mathbf{1}_n^T$)
        $\hat{\mathbf{x}}, \boldsymbol{\psi} \leftarrow$ POSTERIORUPDATE($\hat{\mathbf{x}}, \boldsymbol{\psi}, y, n, \boldsymbol{\Sigma}_{\mathbf{w}}$)
      $\mathbf{x}_m \leftarrow$ GAUSSIANSAMPLE($\hat{\mathbf{x}}, \boldsymbol{\psi}$)
      $u(n) \leftarrow$ Z($\mathbf{x}_m$) ▷ Compute query value
    $\nu(n) \leftarrow$ VAR($\mathbf{u}$) ▷ Sample variance
  return $\arg\min \nu$

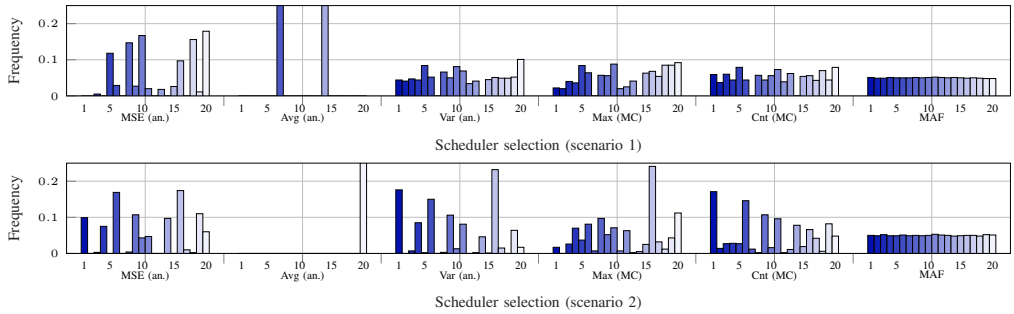


Fig. F.2: Sensor selection frequency for the considered policies.

The operation of the scheduler is specified in Alg. 1: in order to estimate the expected MSE when selecting each sensor n, it samples from the posterior distribution of the query. First, the scheduler performs the prior update step from (F.2) and (F.3), then it draws an outcome to simulate the transmission, with failure probability ε_n. If the simulated transmission was successful, an observation is randomly drawn from a Gaussian distribution with mean $\mathbf{1}_n \hat{\mathbf{x}}$ and variance $\mathbf{1}_n \mathbf{S}(t) \mathbf{1}_n^T$, and the posterior update is performed. We then have the parameters $\hat{\mathbf{x}}$ and $\boldsymbol{\psi}$ of the multivariate Gaussian belief distribution of the state, from which we can draw a sample $\mathbf{x}_m$ to compute $u_m = z(\mathbf{x}_m)$. The sample variance over vector $\mathbf{u}$ is then our estimate of $\mathbb{E}\left[\nu_z(t) | a(t)\right]$, and we can simply select the sensor that gives the minimum expected MSE.

4 Numerical Evaluation

In the following, we show the effects of the sampling strategy on different statistics by simulation, using a Monte Carlo approach: we generate a synthetic process, then try to estimate it at the BS using the different schedulers. The systems below represent two

highly asymmetric examples, but the strategies we derived are optimal for all observable linear systems. The scenarios are constructed to be stable, i.e., the eigenvalues of the system matrices are all smaller than 1.

4.1 Scenario and Settings

We evolve the system for 100 episodes of 1000 samples each, and the Monte Carlo scheduler computes a total of $M = 1000$ samples for each state. We consider two systems with $N = 20$ sensors, in which the elements of the update matrix $\mathbf{A}$ are known. In the first scenario, the matrix $\mathbf{A}_1$ is given by:

$$A_1^{(i,j)} = \begin{cases} \frac{3}{4}, & \text{if } i = j; \\ -\frac{1}{8}, & \text{if } i \neq j, \mathrm{mod}(i - 2j, 7) = 0, \end{cases} \tag{F.20}$$

where $\mathrm{mod}(m, n)$ is the integer modulo function, and the values are 0 everywhere else. On the other hand, in the second scenario, we have:

$$A_2^{(i,j)} = \begin{cases} \frac{4}{5}, & \text{if } i = j; \\ -\frac{1}{9}, & \text{if } i \neq j, \mathrm{mod}(\lceil i - 2.3j \rceil, 7) = 0. \end{cases} \tag{F.21}$$

The other parameters are the same in both scenarios. We also have $\mathbf{\Sigma_w} = \mathbf{I}$, while the process noise covariance is given by:

$$\Sigma_v^{(i,j)} = \begin{cases} \frac{11+\mathrm{mod}(i,10)}{5}, & \text{if } i = j; \\ 1, & \text{if } i \neq j, \mathrm{mod}(i - j, 6) = 0. \end{cases} \tag{F.22}$$

Sensors with higher indices will have a slightly higher variance. The transmission error probabilities are $\varepsilon_n = 0.02 \left\lceil \frac{n-1}{10} \right\rceil$. The filter is initialized at step 0 with state $\hat{\mathbf{x}}(0) = \mathbf{x}(0) = 0$, and $\boldsymbol{\psi}(0) = \mathbf{I}$.

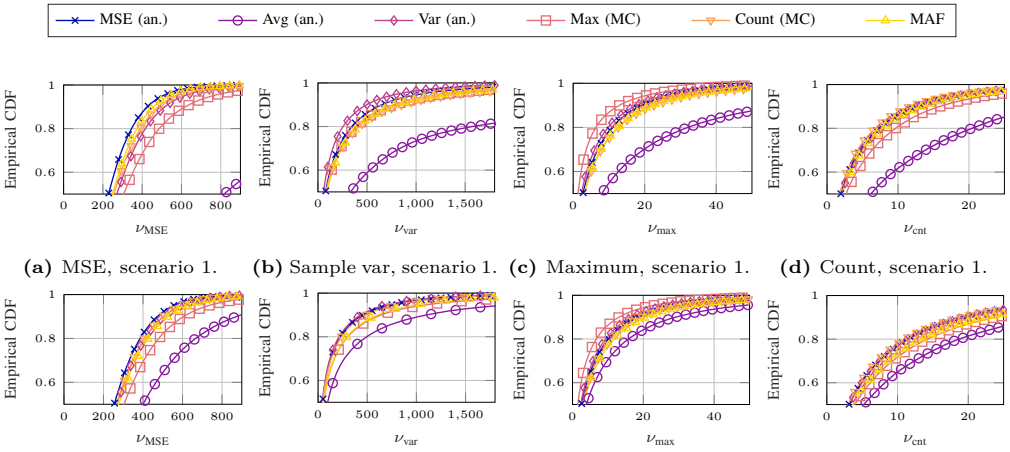
In addition to the baseline scheduler, we also consider the well-known Maximum Age First (MAF) scheduler as a benchmark. If we denote the age of the last received packet from sensor n as Δ_n, the scheduler always picks the sensor with the highest age:

$$a^*_{\mathrm{MAF}}(t) = \underset{n \in \{1,\ldots,N\}}{\arg\max}\ \Delta_n(t). \tag{F.23}$$

Finally, we consider four different summary statistics, as well as the state MSE: aside from the sample mean and variance, we consider the maximum and count statistics, denoted as $z_{\max} = \max_n x_n(t)$ and z_{cnt}, which is given by:

$$z_{\mathrm{cnt}}(\mathbf{x}) = \sum_{n=1}^{N} \mathbb{1}(x_n(t) - a)\mathbb{1}(b - x_n(t)), \tag{F.24}$$

where $\mathbb{1}(x)$ is the step function, equal to 1 if $x \geq 0$ and 0 otherwise, and the count interval $[a, b]$. In other words, the count statistic is a simple count of the number of state components that are within $[a, b]$, which we set to $[-5, 5]$.



(a) MSE, scenario 1. **(b)** Sample var, scenario 1. **(c)** Maximum, scenario 1. **(d)** Count, scenario 1.

(e) MSE, scenario 2. **(f)** Sample var, scenario 2. **(g)** Maximum, scenario 2. **(h)** Count, scenario 2.

Fig. F.3: Estimation error distributions with the different strategies.

4.2 Results

We can first look at the choices of the schemes aimed at each target metric in one of the episodes for each scenario, shown in Fig. F.2. As expected, the MAF scheduler selects sensors with a similar frequency in both scenarios: sensors with an index over 10 are selected slightly more frequently, as transmission errors occur more often, but the difference is small. On the other hand, the average scheduler only selects two sensors, 7 and 14, in the first scenario, and only the last, sensor 20, in the second: this holds throughout all episodes, independently from the state of the system. In the first scenario, alternating between these two sensors gives the best estimate of the overall average, as the state of each of these two sensors only depends on the other's. In the second scenario, no sensor is isolated, but sensor 20 is the one that affects the average the most. The average scheduler then gets the best estimate it can for the other values, concentrating on these sensors and actually getting a better average performance. Naturally, this results in a significantly worse performance when looking at any other summary statistic. We also remark that all other policies excluding MAF never choose sensors 7 and 14 in the first scenario: as errors compound for most of these summary statistics, it does not make sense to choose isolated sensors, as sensors that are more correlated to their neighbors have a better chance to reduce the overall error. We can also note that, in the second scenario, the MSE, count, and sample variance schedulers often make similar choices, while they do not in the first scenario: this similarity is purely due to the specific features of the system, and cannot be relied upon for design.

The CDFs of the quadratic estimation errors $\nu_z(t)$ obtained for the various summary statistics are shown in Fig. F.3. As expected, optimizing the scheduling for a given

summary statistic can reduce the error on it, for all the considered statistics. We did not plot the average statistic, as all policies had a similar performance, although, as expected, the average scheduler performed best for that statistic. However, π^*_{avg} was almost always the worst policy when looking at other summary statistics, often by a wide margin, in both scenarios, for the reasons we explained above. The maximum scheduler π^*_{max} was also noticeably worse when looking at the state MSE or the interval count, as it tended to pick sensors with a high value, often accepting a larger error on other components of the state. As the maximum value was almost always far over the interval boundaries, the count statistic was also negatively affected. Finally, the similarity in the behavior of the count, sample variance, and the MSE scheduler in the second scenario has a similar performance, as Fig. F.3e-h show.

In general, the average and count statistics tend to be relatively insensitive to the scheduling policy used, with most policies showing similar results: in both cases, estimation errors tend to compensate, and the error is relatively low. On the other hand, the gain from using the appropriate scheduling strategy is clearly noticeable when looking at the MSE and at the maximum statistic. In these cases, individual components of the state can have a disproportionate effects, and errors tend to compound rather than compensate each other. In general, while never being the optimum, the MAF scheduler is also never the worst, as it is a purely AoI-oriented approach that does not consider the specific definition of VoI.

5 Conclusion

In this letter, we have considered the optimization of a sensor polling strategy, using different statistics to define the VoI. The difference between the policies can be important, as errors tend to compensate each other in some cases and compound for other statistics, leading to different choices of sensors. Naturally, one-step optimization is a limited approach, and we plan to consider more complex schemes which can take long-term effects into account, as well as different statistics over the same process. Energy consumption is also another important metric, and we plan to compare VoI-based strategies to energy-efficient ones and try to find a balance between them.

References

[1] A. A. Soderlund and M. Kumar, "Optimization of multitarget tracking within a sensor network via information-guided clustering," *J. Guidance, Control, Dynamics*, vol. 42, no. 2, pp. 317–334, Feb. 2019.

[2] A. Awad, A. Mohamed, C.-F. Chiasserini, and T. Elfouly, "Distributed in-network processing and resource optimization over mobile-health systems," *J. Netw. Comput. Appl.*, vol. 82, pp. 65–76, Mar. 2017.

[3] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Conf. on Computer Communications (INFOCOM)*. IEEE, Mar. 2012, pp. 2731–2735.

[4] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 109–117.

[5] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, vol. 42, no. 2, pp. 251–260, Feb. 2006.

[6] A. Hashemi, M. Ghasemi, H. Vikalo, and U. Topcu, "Randomized greedy sensor selection: Leveraging weak submodularity," *IEEE Trans. Autom. Control*, vol. 66, no. 1, pp. 199–212, Mar. 2020.

[7] Z. Li, L. Zhang, Y. Cai, and H. Ochiai, "Sensor selection for maneuvering target tracking in wireless sensor networks with uncertainty," *IEEE Sensors J.*, Dec. 2021.

[8] R. Duan, J. Du, J. Ren, C. Jiang, Y. Ren, and A. Benslimane, "VoI based information collection for AUV assisted underwater acoustic sensor networks," in *Proc. Int. Conf. Commun. (ICC)*. IEEE, Jun. 2020.

[9] S. M. Hoseyni, F. Di Maio, and E. Zio, "VoI-based optimal sensors positioning and the sub-modularity issue," in *Proc. Int. Conf. Syst. Rel. Safety (ICSRS)*. IEEE, Nov. 2019, pp. 148–152.

[10] F. Chiariotti, J. Holm, A. E. Kalør, B. Soret, S. K. Jensen, T. B. Pedersen, and P. Popovski, "Query age of information: Freshness in pull-based communication," *IEEE Trans. Comm.*, 2022.

[11] A. Froytlog *et al.*, "Ultra-low power wake-up radio for 5G IoT," *IEEE Comm. Mag.*, vol. 57, no. 3, pp. 111–117, Feb. 2019.

[12] J. Shiraishi and H. Yomo, "Wake–up control for wireless sensor networks collecting top–k data with temporal correlation," in *Proc. 92nd Veh. Tech. Conf. (VTC2020-Fall)*. IEEE, Nov.

[13] Y. Huang, W. Yu, E. Ding, and A. Garcia-Ortiz, "EPKF: Energy efficient communication schemes based on Kalman filter for IoT," *IEEE Internet of Things J.*, vol. 6, no. 4, pp. 6201–6211, Feb. 2019.

[14] J. Wang, R. Zhu, and S. Liu, "A differentially private unscented kalman filter for streaming data in iot," *IEEE Access*, vol. 6, pp. 6487–6495, Jan. 2018.

[15] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960.

[16] J. Humpherys, P. Redd, and J. West, "A fresh look at the Kalman filter," *SIAM review*, vol. 54, no. 4, pp. 801–823, Nov. 2012.

[17] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. USA: Prentice-Hall, Inc., 1993.

[18] A. M. Mathai and S. B. Provost, *Quadratic forms in random variables: theory and applications*. Dekker, 1992.

[19] D. Luengo, L. Martino, M. Bugallo, V. Elvira, and S. Särkkä, "A survey of Monte Carlo methods for parameter estimation," *EURASIP J. Adv. Signal Process.*, vol. 2020, pp. 1–62, Dec. 2020.

Paper G

Goal-Oriented Scheduling in Sensor Networks with Application Timing Awareness

Josefine Holm, Federico Chiariotti, Anders E. Kalør, Beatriz Soret, Torben B. Pedersen, and Petar Popovski

The paper has been submitted to the
IEEE Transactions on Communications 2022.

The layout has been revised.

Abstract

Taking inspiration from linguistics, the communications theoretical community has recently shown a significant recent interest in semantics and pragmatics (effectiveness) of communication. In this paper, we treat the problem of pragmatic (goal-oriented) communication, i.e., how to transmit the most relevant information for the receiver in order to attain a certain goal, taking into account timing constraints and the shared context. We capture the goal-oriented aspect through the metric of VoI, which is aware of the measured process as well as the timing constraints. However, the most common definition of VoI in the literature is extremely restrictive, using the MSE of the whole system state as a value function, regardless of the relevance of individual components or the needs of specific applications. In this work, we formulate and solve a pragmatic scheduling problem that considers: (1) different summary statistics, i.e., value functions of the state, and (2) a diversified query process on the client side, expressed through the fact that different applications may request different functions of the process state at different times. A query-aware Deep Reinforcement Learning (DRL) solution based on statically defined VoI can outperform naive approaches by 15-20% in certain scenarios.

1 Introduction

Semantic and goal-oriented communications [1] aim to go beyond the traditional domain of communication theory towards optimizing communication systems with respect to a specific task or goal. In [2], Shannon and Weaver talk about the *semantics* and *effectiveness* levels of the communication problem. *Semantic communication* corresponds to the transmission of the most meaningful information for the given context. Following the nomenclature in linguistics, we denote *effectiveness* by the term *goal-oriented* or *pragmatic* communication. In this sense, goal-oriented communication is about transmitting the most relevant information for the receiver in order to attain a certain goal, taking into account both timing constraints and the shared context, which acts as an implicit information channel between the transmitter and receiver.

This new perspective is crucial in the context of the Industry 4.0 and Industrial Internet of Things (IIoT) paradigms, which aim at automating manufacturing and industrial processes in a flexible and easily reconfigurable fashion. A representative scenario is the remote estimation of the state of a system by a distributed network of low-power sensors, a classical IoT problem [3]. The recently proposed VoI metric [4] represents a theoretical tool to model the pragmatics of the monitoring application, as it can seamlessly integrate the timing performance of the communication network with the underlying estimation [5].

However, VoI is often defined in a static manner: the value function is the MSE of the system state, and there is an application constantly monitoring the process with a

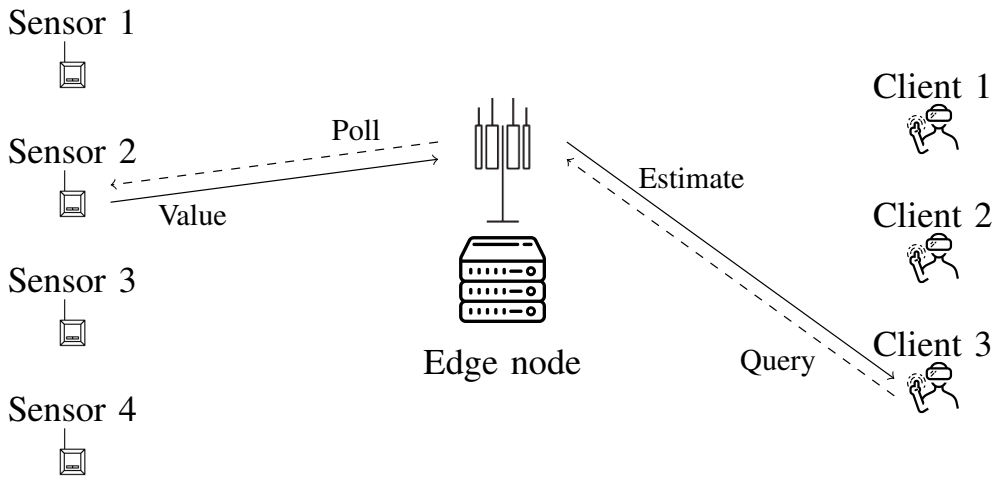
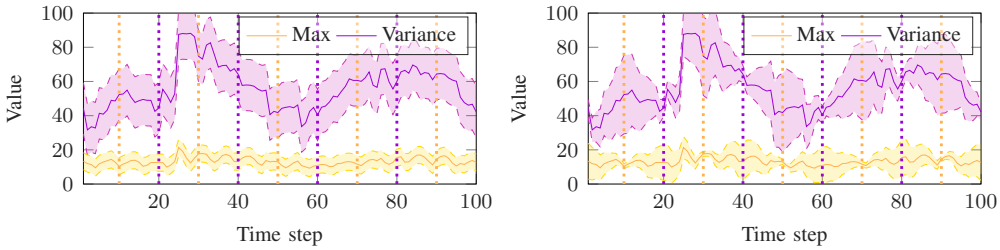


Fig. G.1: Illustration of the considered monitoring scenario, where an edge node collects information from sensors to reply to queries from clients. Queries are indicated by dashed lines, while their responses are solid lines.

faster pace than the updates. In many IoT scenarios, this is not true, as there may be *multiple* clients monitoring the same process through an edge node or gateway. Each client may be potentially interested in a different function of the system state at different times [6], as shown in Fig. G.1. The publish/subscribe model is already the standard for IoT applications [7], with the edge node acting as a message broker. In the system model in the figure, a Mobile Edge Computing (MEC)-enabled base station polls a set of sensors, which respond with their latest measurements. The edge node uses the data to estimate the overall state of the process measured by the sensors, and receives *queries* from client applications, which might be different functions of the state, e.g., the highest value among all sensors, or the number of sensors measuring values in a certain range [8].

This setup has a clear timing context for defining goal-oriented communication: the edge node should optimize its scheduling strategy to answer queries accurately, taking into account both the nature of the estimation process and the query process. The optimal VoI scheduling will then consider not only the *accuracy* of the estimate, i.e., the semantic value of the updates, but also *when* the estimation of a particular function of the state will be needed, i.e., the pragmatic relevance of the information to the receiver. Fig. G.2 shows such an example: a simple strategy that aims at maximizing VoI can maintain an approximately constant error when tracking both the maximum recorded value and the sample variance among the sensors, balancing between the two types of queries. However, an awareness of the query instants, indicated by the vertical dotted



(a) VoI-based sensor scheduling.

(b) Query-aware sensor scheduling.

Fig. G.2: Example of the effects of query-aware VoI scheduling. The lines and colored areas represent the query response value and its confidence interval, respectively, and the vertical dotted lines represent the query times (purple for Variance and orange for Max).

lines in the figure, enables the system to perform better: the error on the value of the maximum, represented by the yellow shaded area, can increase right after a query, but needs to be low when the information is necessary. As queries arrive periodically every 10 time steps, and alternate between the two types, the system should try to minimize the error on the maximum before that type of query, and a sensor with a very low value should not be polled, even if its value would greatly improve the overall estimation of the state. The scheduler should then switch to minimizing the error on the sample variance after replying. As the figure shows, each shaded area is at its narrowest at the moment a query of that type arrives, minimizing the overall error of each response.

In this work, we formulate a pragmatic[1] scheduling problem, considering both the process and the queries from multiple client applications. We define the problem as a MDP, assuming that the edge node has the full statistical knowledge of the process, and solve it with DRL, as the continuous state space prevents us from using simpler methods such as tabular Q-learning. The contributions of the paper are the following:

- We rigorously model the pragmatic sensor scheduling problem with multiple clients and different query functions and query generation processes and find a closed-form solution in a simple example, showing that the optimal policies for different queries can be extremely different and that using the wrong one can lead to significant performance loss;
- We propose a DRL scheduling solution and estimate its computational complexity, concluding that it is executable in real time even on an embedded edge processor;
- We provide simulation results and comparisons with static VoI policies and the traditional MAF policy, which minimizes the average AoI of the system, for a scenario in which two query types are present, i.e., a max query asking the maximum

[1] In the remainder of this paper, we use the terms *pragmatic* and *goal-oriented* interchangeably.

value among the state components and a count range query asking how many state components have a value within a given interval. The performance of the system is strongly dependent on the queries that arrive to it, as the error on the queries depends on the exact function of the state that they request.

The rest of the paper is organized as follows: first, we discuss the state of the art on the topic in Sec. 2. We then define the scheduling problem in Sec. 3, and model it as an MDP in Sec. 4, which also describes the proposed DRL solution. Then, Sec. 5 describes the simulations comparing the proposed scheme to state-of-the-art policies and their results. Finally, Sec. 6 concludes the paper and presents some possible avenues for future work.

2 Related Work

During the last decade, AoI [9] and the associated metrics have received a significant attention in the communication engineering community. Most works in the literature study the average AoI in queues and networks of queues, using basic queuing theory to compute analytical performance curves [4]. However, the original definition of the AoI can result in suboptimal outcomes if some conditions are not met, and similar metrics that can extend the definition to more general scenarios and applications have been defined.

Firstly, AoI does not need to be linear: some recent works generalized the definition to measure any non-decreasing function of the age [10, 11], leading to different considerations in AoI optimization, as higher ages are penalized more or less depending on whether the aging function is super- or sub-linear. This can be further generalized by taking into account the actual value of the process tracked by the updates: since AoI is a proxy metric for the evolution of the tracking error over time, considering that error directly leads to better performance. The Age of Incorrect Information (AoII) [12] mixes a linear timing penalty with a multiplier based on the error in a discrete system, while VoI [13] directly measures the error (either real or expected) on the estimation process.

The adaptive scheduling of sensors in IoT scenarios with the goal of minimizing the AoI or related metrics is a well-studied problem in the literature. The scheduling problem can be formulated both for multiple sources, in which case it involves balancing the ages of the different sources while avoiding interference [14], or for a single source with resource constraints: usually, these constraints are in the form of limited energy availability or enforced duty cycles. The most interesting works in this sense consider the VoI, measured as the expected tracking error of a Kalman filter [15, 16]. In general, the constraint on the accuracy is due to a communication bottleneck, which occurs due to limited bandwidth and energy, such that sensors need to reduce their transmissions as much as possible. Other scenarios in which VoI is used are data muling applications, in which drones, robots, or underwater vehicles need to physically move close to sensors

to collect the information [17], and sensor placement problems, in which the issue is not to schedule transmissions, but rather to design the network to maximize accuracy and minimize cost [18]. Our own previous work [8] extends the definition of VoI from the MSE of the state to arbitrary functions, presenting a one-step optimal scheduling procedure. Another interesting development involves the modeling of the state of each sensor as a Markov chain, posing the polling problem as a Partially Observable Markov Decision Process (POMDP) [19] to identify sensors reporting abnormal values with the minimum energy expenditure [20].

Another assumption that has been central to the AoI literature is that information is always relevant, and that the application that tracks the process is always active. We can relax this assumption by considering a *query process*, as we did in our previous work [6], in which we defined the QAoI: this metric is a sub-sampling of the AoI, only considering the instants when a query arrives from the application to be relevant for the optimization. A similar approach was adopted in defining the EAoI [21], with a slightly different set of assumptions, and our theoretical results were extended in [22, 23], which showed the different outcome of the AoI and QAoI minimization problems under an update-or-wait model. Another work also models requests in the optimization function [24], but it only deals with memoryless request processes, which (as we will describe in the introduction) lead to a solution that is equivalent to standard AoI minimization. The extended version of that paper [25] considers more complex scenarios with partial battery knowledge, but still uses the same memoryless request model. Finally, a recent work by Xu *et al.* [26] also considers a memoryless request process, but considers a mix between traditional AoI and query-aware metrics. By only tracking the AoI when a query arrives from the application, the communication system considers not only the freshness of the received information, but also when it is needed: if, for example, the application works over discrete time intervals, transmitting more data close to the next query can reduce the bandwidth and energy usage, while maintaining the same or better accuracy from the application's point of view.

The problem of value-oriented scheduling has also been approached in the distributed control: if the agents have communication capabilities, the most valuable piece of information is the one that will allow them to improve their performance in the task. The Urgency of Information (UoI) metric [27] directly considers how much an update would affect a known linear controller. If we consider more complex Multi-Agent Reinforcement Learning (MARL) agents, the problem is more complex [28], as the communication policy is implicitly learned by the agents while they converge to the optimal control policy, and this approach has only been successful in simple problems [29] or with only one supporting agent communicating to a primary one [30]. For a more thorough review of the cooperative MARL literature, we refer the reader to [31].

This work combines and extends some of the ideas on VoI sensor scheduling and query awareness, as well as concepts from the semantic communications literature, by considering a system with multiple queries arriving at different times, each of which

requires different information on the state of the tracked process, represented by a different VoI function. To the best of our knowledge, this is the first work to consider this complete system model, and a significant step forward towards full-fledged goal-oriented communications.

3 System Model

We consider a system in which an edge node receives information from a set of N sensors, indexed by $n \in \{1, 2, \ldots, N\}$, and has to respond to *queries* from users in the cloud. Time is divided into slots, denoted as $t = 0, 1, 2, \ldots$, and in each slot, the edge node can send a request to one sensor, and respond to queries from any number of clients. In turn, the sensors observe a linear dynamic system, whose state is denoted as $\mathbf{x}(t) \in \mathbb{R}^M$. The dimensionality of the process state is M, which can be different from the number of sensors N in the general case. The system evolves according to a (potentially time-varying) transition matrix $\mathbf{A}$, with an overlaid error perturbation modeled as a multivariate Gaussian noise:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t-1) + \mathbf{v}(t), \tag{G.1}$$

where $\mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}_M, \mathbf{\Sigma}_v)$. The noise $\mathbf{v}(t)$ is zero-mean, and its covariance matrix is $\mathbf{\Sigma}_v \in \mathbb{R}^{M \times M}$. The sensors then measure a vector $\mathbf{y}(t) \in \mathbb{R}^N$, which represents a linear observation of the state of the system with an added Gaussian measurement noise. We define an observation matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$, which defines the linear function of the state that each function observes:

$$\mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{w}(t), \tag{G.2}$$

where $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}_N, \mathbf{\Sigma}_w)$. The observation noise $\mathbf{w}(t)$ is also zero-mean, with a covariance matrix $\mathbf{\Sigma}_w \in \mathbb{R}^{N \times N}$.

3.1 Remote Kalman Tracking

As the edge node does not know the real state $\mathbf{x}(t)$ of the monitored process, it needs to estimate it. In this work, we use the well-known Kalman filter [32], which is the optimal solution for linear dynamic systems. We assume that the edge node knows the matrices $\mathbf{A}$, $\mathbf{H}$, $\mathbf{\Sigma}_v$, and $\mathbf{\Sigma}_w$, and define vector $\hat{\mathbf{x}}(t) \in \mathbb{R}^M$ as the best estimate of the state available to the edge node. The Kalman filter also outputs a covariance matrix $\boldsymbol{\psi}(t)$, which corresponds to the expected value $\mathbb{E}[(\mathbf{x}(t) - \hat{\mathbf{x}}(t))^T(\mathbf{x}(t) - \hat{\mathbf{x}}(t))]$. We can then provide a recursive formula for updating the *a priori* estimate:

$$\hat{\mathbf{x}}_{t-1}(t) = \mathbf{A}\hat{\mathbf{x}}_{t-1}(t-1) \tag{G.3}$$

$$\boldsymbol{\psi}_{t-1}(t) = \mathbf{A}\boldsymbol{\psi}_{t-1}(t-1)\mathbf{A}^T + \mathbf{\Sigma}_v, \tag{G.4}$$

where the subscript in the estimates indicates the last available observation.

As we stated above, the sensor can request the current value $y_n(t)$ from one sensor per timeslot, whose index is denoted by $a(t)$. We also consider communication errors, modeling the channel between sensor n and the edge node as a PEC with error probability ε_n. Considering the row vector $\mathbf{h}(t) \in \mathbb{R}^M$:

$$\mathbf{h}(t) = \mathbb{1}_{a(t)}\mathbf{H}, \tag{G.5}$$

where $\mathbb{1}_{a(t)}$ is the row vector of length N whose elements are all 0, except for the one with index $a(t)$, which is equal to 1. We can then update the observation function in (G.2), getting the value $y_{a(t)}(t)$, which is the observation transmitted by the polled sensor $a(t)$:

$$y_{a(t)}(t) = \mathbf{h}(t)\mathbf{x}(t) + \mathbb{1}_{a(t)}\mathbf{w}(t) = \mathbb{1}_{a(t)}\mathbf{y}(t). \tag{G.6}$$

We indicate the outcome of the transmission at time t by the Bernoulli random variable $\lambda(t)$, which is equal to 1 if the transmission is successful and 0 otherwise. In the former case, the edge node receives observation $y_{a(t)}(t)$, while in the latter, it receives no observation for this time step. We can then give the Kalman gain row vector $\mathbf{K}(t) \in \mathbb{R}^M$ as:

$$\mathbf{K}(t) = \boldsymbol{\psi}_{t-1}(t)\mathbf{h}(t)^T \left(\mathbf{h}(t)\boldsymbol{\psi}_{t-1}(t)\mathbf{h}(t)^T + \sigma_w^2(t)\right)^{-1}, \tag{G.7}$$

where $\sigma_w^2(t) = \mathbb{1}_{a(t)}\boldsymbol{\Sigma}_w\mathbb{1}_{a(t)}^T$. The update from the *a priori* estimate of the state to the *a posteriori* one is then given by:

$$\hat{\mathbf{x}}_t(t) = \hat{\mathbf{x}}_{t-1}(t) + \lambda(t)\mathbf{k}(t)\left(\mathbf{y}_{a(t)}(t) - \mathbf{h}(t)\hat{\mathbf{x}}_{t-1}(t)\right) \tag{G.8}$$

$$\boldsymbol{\psi}_t(t) = \left(\mathbf{I}_M - \lambda(t)\mathbf{k}(t)\mathbf{h}(t)\right)\boldsymbol{\psi}_{t-1}(t), \tag{G.9}$$

where $\mathbf{I}_M$ is the $M \times M$ identity matrix. We highlight that the *a priori* and *a posteriori* estimates are the same if $\lambda(t) = 0$, as no observation is received by the edge node [33], as the *a priori* estimate is the best estimate that the edge node can obtain with the information it has received.

3.2 The Query Process

We consider a *query* to be a request for either the state $\mathbf{x}(t)$ itself, or the value of a function $z(\mathbf{x}(t))$ of it. The edge node receives queries and responds with an estimate $\hat{z}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t))$ based on the current state of the Kalman filter.

We can consider different types of queries, which can correspond to different functions of the state. A classic query is the sample average among the measured values, but non-linear functions and even order statistics can often be useful in industrial settings. For example, the number of state components that are within their normal operation parameters, or the maximum among the state components, can be helpful to trigger safety conditions and shut down machines or raise a warning to the operators. As

another example, the sample variance can be useful when monitoring the strain on different components of a building or a structure, such as a bridge.

The temporal element of the query process can be modeled as a Markov chain. We assume that each client c follows an independent Markov chain with a state at time t $q_c(t) \in \mathcal{Q}_c$, with a known transition matrix $\mathbf{T}_c$. Each client c always requests the same function z_c anytime its Markov chain is in a subset of states, which we denote as $\tilde{\mathcal{Q}}_c$. Naturally, the state of each client is unknown to the edge node, which can only know which clients are currently subscribed and when did they send their last query. In some cases (e.g., periodic queries), this information is sufficient to predict the next query perfectly, as we will discuss below, but in the general case, the information about the query process available to the edge node entails some randomness and uncertainty.

3.3 Responding To Queries

The objective of the edge node is to respond to queries as accurately as possible, i.e., to minimize the error of its responses. The MSE for client c is defined as:

$$\mathrm{MSE}_{z_c} = \mathbb{E}\left[\left(\hat{z}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t)) - z(\mathbf{x}(t))\right)^2\right]. \tag{G.10}$$

The edge node can act in two ways to minimize the MSE: the first is to optimally use its knowledge of the state by using an MMSE estimator to obtain $\hat{z}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t))$, and the second is to poll sensors according to the expected VoI of their readings, i.e., schedule the sensor that can help the most in reducing the MSE of the next queries. The former problem is relatively simple, while the latter will be tackled in Sec. 4.

We can give the definitions and MMSE estimators for some of the most intuitive queries as follows:

1. *State*: in this case, the request is for the direct value of $\mathbf{x}(t)$. As the Kalman filter is the MMSE estimator as long as the system is correctly identified, the response of the node will be $\hat{\mathbf{x}}_t(t)$, and the MSE will be $\mathrm{tr}(\boldsymbol{\psi}_t(t))$, where $\mathrm{tr}(\cdot)$ is the matrix trace operator;

2. *Sample mean*: in this case, the function that the client requests to the edge node is the sample average, represented by:

$$z_{\mathrm{avg}}(\mathbf{x}(t)) = \frac{1}{M}\sum_{m=1}^{M} x^{(m)}(t), \tag{G.11}$$

where $x^{(m)}(t)$ is the m-th element of $\mathbf{x}(t)$. The optimal response is simple: as $\mathbf{x}(t) - \hat{\mathbf{x}}(t)$ is a multivariate Gaussian variable with zero mean and covariance matrix $\boldsymbol{\psi}_t(t)$, the MMSE estimator is simply given by $z_{\mathrm{avg}}(\hat{\mathbf{x}}_t(t))$, and its MSE is

given by:

$$\mathrm{MSE}_{\mathrm{avg}}(t) = \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}\psi_t^{(i,j)}(t), \tag{G.12}$$

where $\psi_t^{(i,j)}(t)$ is element (i,j) of the covariance matrix $\boldsymbol{\psi}_t(t)$;

3. *Sample variance*: in this case, the sample variance is computed as:

$$z_{\mathrm{var}}(\mathbf{x}(t)) = \frac{1}{M-1}\sum_{m=1}^{M}\left(x^{(m)}(t) - z_{\mathrm{avg}}(\mathbf{x}(t))\right)^2. \tag{G.13}$$

In order to derive the MMSE estimator, we express the sample variance as the following quadratic form:

$$z_{\mathrm{var}}(\mathbf{x}(t)) = \frac{1}{M-1}\mathbf{x}(t)^T\mathbf{Q}\mathbf{x}(t), \tag{G.14}$$

where $\mathbf{Q} = \mathbf{I} - 1/M$. Since the difference between the real and estimated states is multivariate Gaussian, we can compute the MMSE estimator using the method from [34]:

$$\hat{z}_{\mathrm{var}}(t) = \frac{1}{M-1}\left(\mathrm{tr}\left(\mathbf{Q}\boldsymbol{\psi}_t(t)\right) + \hat{\mathbf{x}}_t(t)^T\mathbf{Q}\hat{\mathbf{x}}_t(t)\right). \tag{G.15}$$

In this case, the MSE of the estimator is given by:

$$\mathrm{MSE}_{\mathrm{var}}(t) = \frac{2\mathrm{tr}\left(\mathbf{Q}\boldsymbol{\psi}_t(t)^2\right) + 4\hat{\mathbf{x}}_t(t)^T\mathbf{Q}\boldsymbol{\psi}(t)\hat{\mathbf{x}}_t(t)}{(M-1)^2}. \tag{G.16}$$

The optimal scheduler, and even the MMSE estimator, for more complex queries, with highly non-linear functions, is hard to compute analytically. In the case of order statistics, a closed-form MMSE estimator might not even be achievable [35], as the extreme values of high-dimensional multivariate Gaussian variables are computed only as limiting distributions in the relevant literature. Another common function, which we indeed use in our performance analysis, is the *count range* function, which counts how many components of the state are inside a given interval $[a,b]$. The count range query is defined as:

$$z_{\mathrm{cnt}}(\mathbf{x}(t)) = \sum_{m=1}^{M}\mathbb{1}\left(x^{(m)}(t) \in [a,b]\right), \tag{G.17}$$

where $\mathbb{1}(\cdot)$ is the indicator function, equal to 1 if the condition inside the parentheses is verified and 0 otherwise. Note that the definition of the function $z_{\mathrm{cnt}(\cdot)}$ should include the interval $[a,b]$, but here we omit it for the sake of readability. Furthermore, a and

b are assumed to be fixed for the same query process. We can then define the region $\mathcal{Z}(m)$, which is defined as follows:

$$\mathcal{Z}(m) = \left\{ \mathbf{x} \in \mathbb{R}^M : z_{\text{cnt}}(\mathbf{x}) = m \right\}. \tag{G.18}$$

We can then define the probability that $z_{\text{cnt}}(\mathbf{x}(t))$ is equal to m, corresponding to the integral of the multivariate Gaussian random variable $\mathbf{x}(t) \sim \mathcal{N}(\hat{\mathbf{x}}_t(t), \boldsymbol{\psi}(t))$ in $\mathcal{Z}(m)$:

$$\mathcal{P}(z_{\text{cnt}}(\mathbf{x}(t)) = m) = \int_{\mathcal{Z}(m)} \frac{1}{\sqrt{2\pi^M |\boldsymbol{\psi}(t)}} e^{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}}_t(t))^T \boldsymbol{\psi}^{-1}(t)(\mathbf{x}-\hat{\mathbf{x}}_t(t))} d\mathbf{x}. \tag{G.19}$$

The MMSE estimator for the count range query is then the average over all possible values of m:

$$\hat{z}_{\text{cnt}}(t) = \sum_{m=0}^{M} m\mathcal{P}(z_{\text{cnt}}(\mathbf{x}(t)) = m). \tag{G.20}$$

The corresponding MSE is defined as follows:

$$\text{MSE}_{\text{cnt}}(t) = \sum_{m=0}^{M} (m - \hat{z}_{\text{cnt}}(t))^2 \mathcal{P}(z_{\text{cnt}}(\mathbf{x}(t)) = m). \tag{G.21}$$

The integral in (G.19) can only be computed numerically, and is extremely hard to tabulate. Computing the effect of scheduling each sensor is computationally heavy, and is required to compute the VoI.

However, the MMSE scheduler for any query function can be easily approximated using Monte Carlo methods [36] by drawing samples from the *a priori* distribution. The detailed algorithm for Monte Carlo-based scheduling is given in our previous work [8]. While Monte Carlo estimates are not MMSE, they approach the optimal estimator as the number of samples grows to infinity, at the cost of computational complexity. If we consider S samples, the complexity of one Monte Carlo estimate is $O(SM^2)$.

4 The Scheduling Problem

In the previous section, we defined the system model and determined the optimal estimator for common query functions, along with a Monte Carlo strategy for general functions. However, the most complex problem is not to reply directly to a query, but to consider future queries in a foresighted manner, scheduling sensor transmissions so as to minimize the MSE on future responses. This requires to consider not only the monitored system, but also the query process and the interplay between different query functions. For example, two clients which request the maximum and minimum will need very different parts of the state to be estimated accurately, and balancing between their

needs will be complex. The polling decisions made by the edge node also affect the future state of the Kalman filter, requiring a dynamic strategy.

We can model the scheduling problem for the edge node as a POMDP, in which the edge node must decide which sensor to poll at each time slot. The action space is then simply $\mathcal{A} = \{1, \dots, N\}$, while the state space is more complex. The state of the Kalman filter just before the update, described by $\hat{\mathbf{x}}_{t-1}(t)$ and $\boldsymbol{\psi}_{t-1}(t)$, is included in the state, and so should all the states of the clients: the state space for a system with C clients is then $\mathcal{S} = \mathbb{R}^{M^3} \times \prod_{c=1}^{C} |\mathcal{Q}_c|$. However, the edge node does not know the state $q_c(t)$ of each client, but only the time that has passed since the last query, which we define as $\tau_c(t) \in \mathbb{N}$. We then have an observation space $\mathcal{O} = \mathbb{R}^3 \times \mathbb{N}^C$. The matrices $\mathbf{A}$, $\mathbf{H}$, $\boldsymbol{\Sigma}_v$, and $\boldsymbol{\Sigma}_w$, as well as the error probability vector $\boldsymbol{\varepsilon} = [\varepsilon_n]$ and the query functions z_c, should also be known *a priori* to the edge node, but are not part of the state.

Note that the problem reduces to a fully observable MDP if the time since the last transmission is sufficient to determine the next query, i.e., if the following condition is true:

$$\mathcal{P}(q_c(t+1) \in \tilde{\mathcal{Q}}_c | q_c(t) = q) = \mathcal{P}(q_c(t+1) \in \tilde{\mathcal{Q}}_c | \tau_c(t)), \quad \forall q \in \mathcal{Q}_c, \tau_c(t) \in \mathbb{N}. \tag{G.22}$$

Two special cases of this are the memoryless process, in which the Markov chain only has two states (query and no query), and the deterministic chain with $|\tilde{\mathcal{Q}}_c| = 1$, which leads to a periodic query process. In the general case, the state of the query process depends on external factors (e.g., a human operator), and is not directly knowable by the edge node: if a stochastic transition can lead to a state in which (G.22) is not verified, the problem is partially observable.

The transition probability $P(s, s'|a)$ from one state to the next for a given action is then determined by the Markov chains of each client, along with the Kalman filter equations in (G.8) and (G.9). The final parameter to define the POMDP is then the reward function $r(t)$, which is simply given by the following:

$$r(t) = -\sum_{c \in \mathcal{C}} \alpha_c \mathrm{MSE}_{z_c}(t) \mathbb{1}(q_c(t) \in \tilde{\mathcal{Q}}_c), \tag{G.23}$$

where $\alpha_c > 0$ is a weight parameter representing the relative importance of each client, whose value is given by the system designers, and is thus known *a priori* by the edge node. The reward is always negative, as the objective is to minimize the error on all queries.

We then define a *policy* $\pi : \mathcal{O} \to \Phi(\mathcal{A})$, where $\Phi(\mathcal{A})$ is a probability distribution over the action space $\mathcal{A}$. In other words, the policy maps observed states to the probability of selecting each sensor. We can then define the long-term reward function $R(t|\pi)$:

$$R(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(t) \Big| s_0, \pi\right], \tag{G.24}$$

where $\gamma \in [0, 1)$ is an exponential discount factor. The objective of the scheduling problem is then to find the optimal policy π^*, which maximizes the long-term reward:

$$\pi^* = \underset{\pi:\mathcal{O}\to\Phi(\mathcal{A})}{\arg\max}\ R(\pi). \tag{G.25}$$

The case for $\gamma = 0$ is a special case, in which future steps are never counted, and only performance in the next step matters: this case was solved analytically in our previous work [8].

4.1 A Simple Example: The Effect of Queries on the Optimal Policy

We can first consider a simple example, in which a system with $N = 2$ sensors observes a process with $M = 2$ and needs to reply to a single client (i.e., $C = 1$). The communication is assumed to be error-free, and each sensor m observes an independent binary Markov chain, which changes its state with probability p_m, so that $\mathbf{x}(t) \in \{0, 1\}^M \, \forall t$ and $\mathbf{H} = \mathbf{I}_2$. The transition matrix $\mathbf{T}_m$ is then given by:

$$\mathbf{T}_m = \begin{pmatrix} 1 - p_m & p_m \\ p_m & 1 - p_m \end{pmatrix}. \tag{G.26}$$

We know that the observation of the state is error-free, so after Δ_m steps from the last observation o_m, the *a posteriori* state probability distribution of Markov chain m is given by:

$$P_m(\Delta_m, o_m) = \mathbf{T}_m^{\Delta_m} \begin{pmatrix} 1 - o_m \\ o_m \end{pmatrix}. \tag{G.27}$$

We assume that a query arrives at every step from the client, but define two types of clients with different query functions:

- The *maximum* query returns the maximum between the two values:

$$z_{\max}(\mathbf{x}(t)) = \max_{m\in\{1,2\}} x^{(m)}(t). \tag{G.28}$$

- In this case, the *count range* query counts the number of sensors which have a value equal to 1:

$$z_{\mathrm{cnt}}(\mathbf{x}(t)) = \sum_{m=1}^{2} \mathbb{1}\left(x^{(m)}(t) = 1\right). \tag{G.29}$$

If at least one of the sensors has a value of 1, the value of the other sensor is meaningless for the maximum query; on the other hand, it is still relevant for the count query. We

can compute the maximum likelihood response to each query:

$$\hat{z}_{\max}(\boldsymbol{\Delta}, \mathbf{o}) = 1 - P_{1,0}(\Delta_1, o_1) P_{2,0}(\Delta_2, o_2); \tag{G.30}$$

$$\hat{z}_{\text{cnt}}(\boldsymbol{\Delta}, \mathbf{o}) = P_{1,1}(\Delta_1, o_1) + P_{2,1}(\Delta_2, o_2), \tag{G.31}$$

where $P_{m,i}(\Delta_m, o_m)$ is the *a posteriori* probability that chain m will be in state i, given the latest observation and its age, and $\boldsymbol{\Delta}$ and $\mathbf{o}$ are the vectors of ages and observed values, respectively. We can also compute the MSE for both queries:

$$\text{MSE}_{\max}(\boldsymbol{\Delta}, \mathbf{o}) = P_{1,0}(\Delta_1, o_1) P_{2,0}(\Delta_2, o_2)(1 - P_{1,0}(\Delta_1, o_1) P_{2,0}(\Delta_2, o_2)); \tag{G.32}$$

$$\text{MSE}_{\text{cnt}}(\boldsymbol{\Delta}, \mathbf{o}) = P_{1,0}(\Delta_1, o_1) + P_{2,0}(\Delta_2, o_2) - \left(P_{1,0}(\Delta_1, o_1)\right)^2 - \left(P_{2,0}(\Delta_2, o_2)\right)^2. \tag{G.33}$$

We can note that, if we observe one of the two states and $o = 1$, the response to the maximum query is always correct, as the probability of that component being equal to 0 is 0. In order to maximize the long-term reward from (G.24), we can adopt the classical policy iteration method, as described in [37, Ch. 4] after truncating the POMDP by setting a maximum age $\Delta_{\max}$. The state is then defined as $s = (\boldsymbol{\Delta}, \mathbf{o}) \in \mathcal{S}$. The transitions from one state to the other are extremely simple, and we can easily derive the transition probability $\mathcal{P}(s(t+1)|s(t), \pi(s(t)))$ for each possible combination.

Policy iteration has two steps, called evaluation and improvement. The algorithm is initialized with an approximate value $V_0(s)$ for each state and a policy π_0, which can be set as all zeros. It then repeats the two steps iteratively until the policy converges. In the first step at iteration t, the value function $V_t(s)$ is updated as follows:

$$V_{t+1}(s) = -\text{MSE}(s, \pi_t(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, \pi_t(s)) V_t(s'). \tag{G.34}$$

Naturally, the definition of the MSE depends on the type of query. After the value has been updated for all states, the policy is updated:

$$\pi_{t+1}(s) = \arg\min_{a \in \{1,2\}} -\text{MSE}(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a) V_{t+1}(s'). \tag{G.35}$$

Policy iteration is guaranteed to converge to the optimal policy in finite-state MDPs with finite reward [38].

The results for $p_1 = 0.1$ and $p_2 = 0.2$ are given in Fig. G.3. As Fig. G.3a-d show, the policy is the same for any observation, and only depends on the age of the two measurements, since the MSE of the count query is the same for any observation. The level of uncertainty determines the action: the second component of the state, which can vary more often due to the higher state change probability, is the one that is polled, unless the age of the latest observation of the other component is approximately double. The maximum query has a more complex policy: if the last observations of each

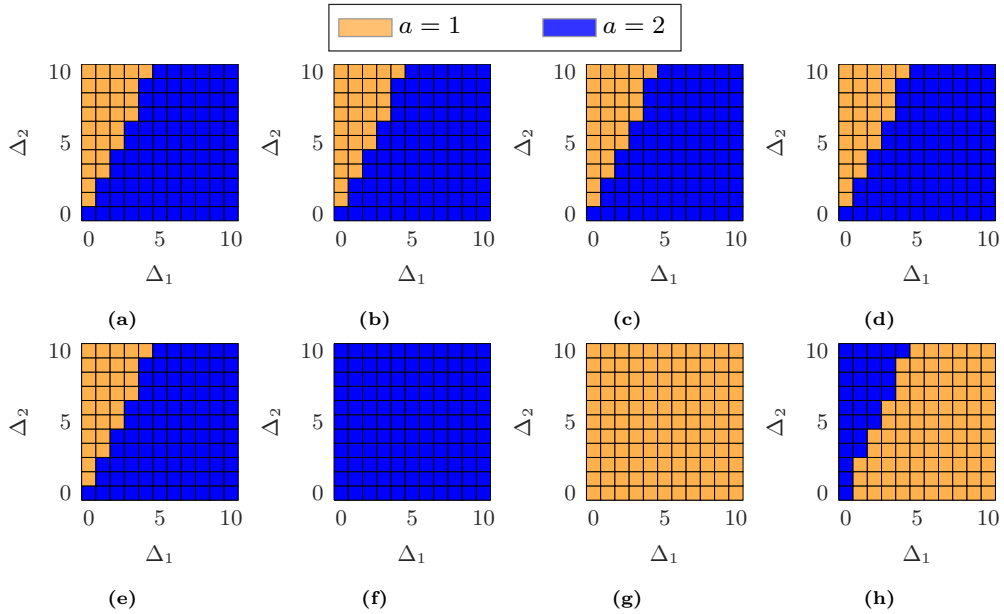


Fig. G.3: Optimal policies with $p_1 = 0.1$ and $p_2 = 0.2$. a–d: Count range query with $\mathbf{o} = (0,0)$, $\mathbf{o} = (0,1)$, $\mathbf{o} = (1,0)$, $\mathbf{o} = (1,1)$. e–h: Max query with $\mathbf{o} = (0,0)$, $\mathbf{o} = (0,1)$, $\mathbf{o} = (1,0)$, $\mathbf{o} = (1,1)$.

component are the same, i.e., $\mathbf{o} = (0,0)$ or $\mathbf{o} = (1,1)$, the policy is the same as for the count range query. On the other hand, if one of the last observations is 1, while the other is 0, the component with value 1 is always polled. This is reasonable, as one observation from a sensor that contains a 1 gives perfect certainty on the overall maximum. Giving a higher priority to the component with the highest probability of being equal to 1 is then beneficial, even if the uncertainty on the other component becomes extremely high.

Naturally, this is only a simple example, and introducing a query process will complicate the system, but it highlights the strong dependence between the function that determines each query and the respective polling policy. While the optimal strategy to minimize the AoI would always poll the sensor with the highest age, and a strategy that minimizes the uncertainty of the count range query (which, in this simple system, is almost equivalent to minimizing the MSE) weighs each sensor's age by the speed of the corresponding process, the strategy for the maximum query actually depends on the current value of each sensor, and is starkly different from the others. Mixing different types of query in the same system will then lead to non-trivial trade-offs, particularly when the functions are highly non-linear.

4.2 Reinforcement Learning Solution and Learning Architecture

While policy iteration has strong convergence guarantees, it is infeasible to use when the state space is large, which is the case for the considered scheduling problem. Instead, we resort to approximate solutions, and consider a Reinforcement Learning (RL) approach to the scheduling problem. RL is a machine learning approach in which an agent learns from experience, updating its estimate of the value function by trial and error. The agent makes decisions and receives immediate rewards from the environment, without any prior knowledge of the reward function or the consequences of actions. For a more thorough introduction to reinforcement learning, we refer the reader to [37].

We implement the Deep Q Network (DQN) architecture [39], which uses a deep neural network to approximate the value function. In order to avoid instability, we need to use a *replay memory* to store the agent experience and select batches of uncorrelated samples. Each batch contains B uncorrelated samples, and each experience sample is a tuple $e = (s(t), a(t), r(t), s(t+1))$. We maintain two neural networks for increased stability: a *target network* and an *update network*. In order to estimate the long-term reward $R(\pi)$ from an experience sample, we use the target network's prediction $Q_t(s, a)$:

$$Q(e) = r(t) + \gamma \max_{a \in \mathcal{A}} Q_t(s(t+1), a). \tag{G.36}$$

The use of the long-term reward estimates to update future estimates follows the well-established bootstrap method, and the use of a greedy update policy follows the Q-learning model implemented by the DQN. The estimates $Q(e)$ are then used as labels for the backpropagation operation on the update network, whose output predictions are used in the action policy to select the next action. The action policy we use implements the well-known softmax function:

$$\pi(s, a) = \frac{e^{\frac{Q_u(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q_u(s,a')}{\tau}}}, \tag{G.37}$$

where the temperature parameter τ is to balance between exploration and exploitation. Lower values of τ make the outcome closer to the greedy policy, as the probability of selecting suboptimal actions decreases, while higher values of τ increase exploration. In any case, exploration with the softmax function is *directed*: actions that are assumed to be highly suboptimal will be picked less frequently, while the agent will prefer actions that have an estimated long-term reward just below the maximum.

The update network is updated at every step with a new batch of samples, while the target network is only updated every U steps by copying the update network's weights. As we stated above, the use of separate target and update networks allows the system to converge, avoiding numerical and stability issues. In the rest of this work, we implement a DQN with 3 layers, whose parameters are given in Table G.1. The first two layers

Table G.1: DQN architecture.

Parameter	Layer 1	Layer 2	Layer 3
Input size	$M^2 + M + C$	$2.5M$	M
Output size	$2.5M$	M	N
Activation function	ReLU	ReLU	ReLU
Dropout	0.1	0.1	0

have a dropout probability $p_d = 0.1$ during the training, and the network is relatively simple, as the input is highly redundant. The hyperparameters above were found after a grid search optimization process.

4.3 Computational Complexity

We can now discuss the computational complexity of the learning solution. The following refers to the complexity of a trained model, i.e., of a single decision on the next action: while training can be performed offline in a simulation environment or even passively on existing data, actions need to be real-time for the system to work, and the time required to make decisions is critical.

If we consider a single layer with ℓ_i inputs and ℓ_o outputs, there are three operations that the network needs to perform to compute each output:

1. Multiply each input ℓ_i by the appropriate weight (equivalent to ℓ_i multiplication operations);
2. Sum all the results (equivalent to ℓ_i sums);
3. Apply the non-linear activation function.

If we consider the activation function as the result of k basic operations, the total number of basic operations for a single layer is then $\ell_o(2\ell_i + k)$. If we consider our architecture as a vector $\boldsymbol{\ell}$ of layer sizes, where the first element is the cardinality of the input (i.e., the observed state) and the last element is N (corresponding to the N possible actions), we have a total complexity of:

$$\mathcal{C}_f(\boldsymbol{\ell}) = \sum_{i=1}^{|\boldsymbol{\ell}|-1} \ell_{i+1}(2\ell_i + k). \tag{G.38}$$

We remark that $\mathcal{C}_f(\boldsymbol{\ell})$ is the total number of basic operations per layer, and as such, If we consider our architecture for $C = 2$ and $N = 20$, which is given above, we have $k = 1$, as the Rectified Linear Unit (ReLU) activation function is extremely simple, and the total number of operations for each step is then $\mathcal{C}_f = 96\,570$. The backpropagation algorithm required to train the neural network has the same complexity as the forward

Table G.2: DQN parameters.

Parameter	Description	Value
γ	Exponential discount factor	0.9
T_e	Time steps in each episode	100
E_{train}	Training episodes	100
E_{test}	Test episodes	10
p_d	Dropout probability	0.1
R_m	Replay memory size	10000
B	Batch size	128
t_{up}	Target net update period	10
L_o	Learning rate optimizer	Adam
L_0	Initial learning rate	10^{-4}

pass, but it must be run for each sample in a training batch [40]. For a training batch size of B, the total complexity for a single training step is then given by:

$$\mathcal{C}_b(\boldsymbol{\ell}) = B\mathcal{C}_f(\boldsymbol{\ell}). \tag{G.39}$$

In our architecture, we have $B = 128$, and consequently, $\mathcal{C}_b = 12\,360\,960$. This number of operations should be entirely within the capabilities of an edge node, as even simple embedded processors can deal with much more complex architectures that require billions of operations in less than 100 ms [41]. As most of the required calculations in training and evaluation are vector operations, each layer might only require a single clock tick on modern processors, particularly when the processor is a GPU or designed for hardware-assisted learning.

5 Simulation Settings and Results

The performance of the RL-based query-aware scheme is verified by Monte Carlo simulation, considering a specific scenario. Its performance is measured in terms of the MSE on its query responses. The evaluation is performed over $E_{\text{test}} = 10$ independent episodes, each of which consists of $T_{\max} = 100$ time steps. The parameters of the DQN agent are the same for all considered scenarios, and are given in Table G.2.

5.1 Scenario and Benchmark Policies

We consider a system with $M = 20$ sensors, each observing a different component of the state $\mathbf{x}(t)$, so that $M = N$ and $\mathbf{H} = \mathbf{I}$. The dynamic system that the edge node observes is defined as follows:

$$\mathbf{A}^{(i,j)} = \begin{cases} \frac{3}{4}, & \text{if } i = j; \\ -\frac{1}{8}, & \text{if } i \neq j \wedge \operatorname{mod}(i - 2j, 7) = 6. \end{cases} \tag{G.40}$$

The edge node knows $\mathbf{A}$, as well as the process and measurement noise covariance matrices, which are given by:

$$\mathbf{\Sigma}_v^{(i,j)} = \begin{cases} \frac{11+\mathrm{mod}(i-1,10)}{5}, & \text{if } i = j; \\ 1, & \text{if } i \neq j, \mathrm{mod}(i-j,6) = 0, \end{cases} \tag{G.41}$$

$$\mathbf{\Sigma}_w = \mathbf{I}. \tag{G.42}$$

The error probability ε_n for each sensor is given by:

$$\varepsilon_n = 0.02 \left\lceil \frac{n}{10} \right\rceil. \tag{G.43}$$

We consider a case with $C = 2$ clients with the same importance, i.e., $\alpha_1 = \alpha_2 = 1$.Client 1 requests a *count range* query, i.e., the number of sensors whose value is in the interval $[-5, 0]$:

$$z_1(\mathbf{x}(t)) = \sum_{m=1}^{M} \mathbb{1}\left(-5 < x^{(m)}(t) < 0\right). \tag{G.44}$$

Client 2, on the other hand, makes a *maximum* query, i.e., requests the highest value in the vector $\mathbf{x}(t)$:

$$z_{\max}(\mathbf{x}(t)) = \max_{m \in \{1,\ldots,M\}} x^{(m)}(t). \tag{G.45}$$

It is possible to add more clients with other queries and varying importance. This adds one parameter (i.e., the time since the last query from that client) to the DQN, but the problem is not guaranteed to scale: the added complexity necessarily makes the training longer, requiring an adjustment to the exploration and learning profiles as well.

We consider 5 different benchmarks for the query-aware policy:

- *MAF*: The MAF policy, which minimizes the average AoI of the system regardless of the value of sensors' readings. This legacy approach represents a value-neutral lower bound;
- *Cnt*: The one-step optimal policy for client 1, which follows the procedure from [8] to minimize the MSE of the count range query in the current step;
- *Max*: The one-step optimal policy for client 2, which does the same for the maximum query;
- *RL (Cnt)*: The foresighted policy learned by a RL agent with $\alpha_2 = 0$, which only minimizes the MSE of the response to the count range query;
- *RL (Max)*: The foresighted policy with $\alpha_1 = 0$, which does the same for the maximum query.

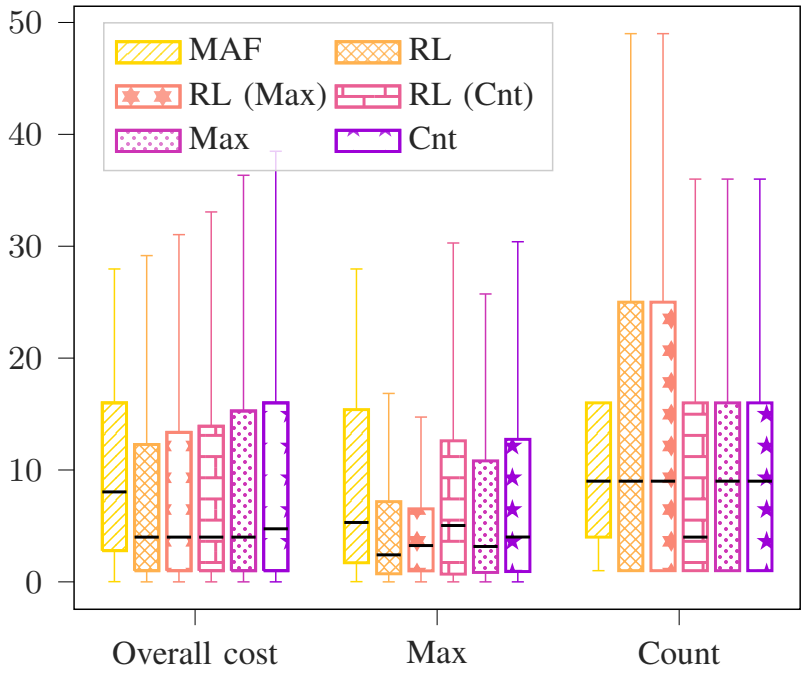
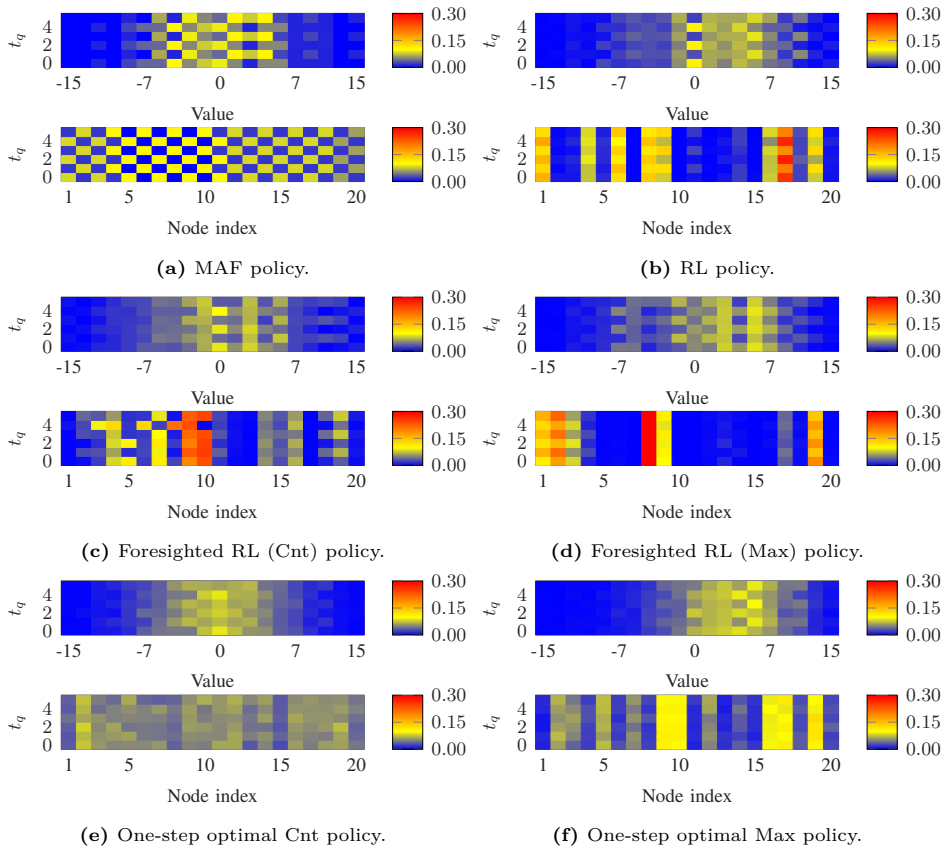


Fig. G.4: MSE cost of the different policies for both types of query in the periodic query scenario.

We also consider two different query processes, both of which are observable by the edge node, slightly simplifying the problem:

- *Periodic queries*: queries are generated every $T_q = 6$ steps. In this case, the Markov chain is deterministic, going from state 0 (in which a query is generated) to state 1 with probability 1, then increasing until 5, after which the chain goes back to 0 with probability 1;
- *Memoryless queries*: in this case, the Markov chain only has 2 states, and the rows of the transition matrix are identical. The time between two subsequent queries is geometrically distributed, with an expected value $\mathbb{E}[T_q] = 6$ steps.

We consider three combinations of these query processes: the case in which both clients have periodic queries, the case in which they both have memoryless queries, and the mixed case in which client 1 has periodic queries, while client 2 follows a memoryless process.



(a) MAF policy.

(b) RL policy.

(c) Foresighted RL (Cnt) policy.

(d) Foresighted RL (Max) policy.

(e) One-step optimal Cnt policy.

(f) One-step optimal Max policy.

Fig. G.5: Colormap representing the histogram of choices, in terms of sensor value and index, for each of the policies. The vertical axis represents the time t_q (count range queries arrive at time 0).

5.2 Periodic Query Scenario

In this subsection we show results for the case where both queries arrive periodically, each with a period of 6 steps. The two queries are out of sync, with the max query starting at 0 and the count range query starting at time 2.

This is the easiest case for the edge node, as queries are generated deterministically and the optimal policy can act on deterministic knowledge of the query pattern. Fig. G.4 shows a boxplot of the MSE for both types of queries, as well as the overall cost: we can note that the RL policy considering both types of queries obtains a lower cost than all others (as shown in the group on the left of the figure), with a much lower average and only slightly worse performance at the 95th percentile than an AoI-oriented approach.

In particular, the choice made by the RL policy is to privilege the max query, with results that end up being similar to only optimizing for it. All other approaches tend to reduce the MSE of the count range query more, although they end up having a higher error on the max query. The count range query is penalized by the fact that it arrives only 2 slots after the max query: reducing its MSE would require losing accuracy in the response to the max query, increasing the overall cost. On the other hand, the 4 slots between a count range query and the subsequent max query allow the RL policy to improve the accuracy significantly. The effect of the discount factor γ is also important: since a count range query arrives 2 slots after the max query before it, and $\gamma = 0.9$, its MSE only accounts for 81% of the reward for the steps before the max query. A higher value of γ, or a different weighting of the two query types by adapting α_1, would produce a more balanced outcome.

We can also note that, in this case, the other RL-based policies outperform their greedy versions on the metric that they optimize for, but no such pattern exists for the other type of query, which these policies completely disregard. As noted in previous works on VoI, the AoI-based approach taken by the MAF policy provides a middle ground for performance, never failing too badly by polling all sensors equally.

The choices made by the various policies can be analyzed more in depth by considering the distribution of the sensors that are polled. Fig. G.5 shows two colormaps for each policy, in which the y-axis represents the step in each query period, i.e., the index of the slot modulus 6. As a reminder, the maximum query is generated at $t_q = 0$ and the count range query is generated at $t_q = 2$. The two colormaps differ by the value represented on the x-axis: in the first one, the x-axis represents the value $x^{(m)}(t)$ measured by the chosen sensor, while in the second, the value is simply the index of the sensor. The color of each cell represents the empirical probability of each combination in our test episodes.

We can first look at the MAF policy, in Fig. G.5a: the distribution of values is almost symmetrical, and values between -5 and 5 are polled with approximately the same frequency. On the other hand, the index colormap shows a checkerboard pattern, caused by the round robin-like pattern of updates (which is shifted by 2 steps at every cycle, as N is not a multiple of the query period). The RL policy has a different pattern: we can note that in even time slots, corresponding to the query instant, the distribution of values is bimodal: sensors whose value is close to 0 are polled very often, as are sensors whose value is very high, between 7 and 12. These two peaks correspond to the two queries: values close to 0 are at the edge of the interval that is relevant for the count range query, while very high values are obviously interesting for the max query. The indexes of the sensors that are polled are also much more concentrated: sensors 1, 6, 8, and 17 are polled extremely often, while other sensors are rarely polled: this is due to the nature of the problem, as some sensors are more valuable to answer the queries due to the evolution of the state.

The two single-query RL policies, whose choices are represented in Fig. G.5c-d, can

further shed light on the behavior of the joint policy: we can easily see that some of the nodes that are often polled by RL are also polled by its Max and Cnt versions, and that the two peaks in the distribution are close to a superposition of the two peaks of RL (Max) and RL (Cnt).

Finally, we can note that the one-step greedy policies, shown in Fig. G.5e-f, do not have any dependence on the time step, as they are unaware of the query process: the basic features, such as the Cnt policy choosing values clustered around 0 and the Max policy choosing values on the highest end of the range, are maintained, but the policies are inherently noisier than their RL-based versions, which can exploit their knowledge of the query process to improve performance at the right moment.

5.3 Geometric Query Arrival

We can consider a second scenario, in which at each step a query of either type is generated with probability 1/6. The average frequency of queries is the same as for the previous scenario, but instead of a deterministic, periodic sequence, queries follow a memoryless random process with geometrically distributed inter-query times. We remark that queries of both types may arrive to the edge node at the same time, and that in this case, no knowledge is available at the edge node: as the query process is memoryless, knowing the arrival times of past queries provides no information on future query arrivals. In this case, the advantage of a query-aware system is naturally diminished. The time since the last query of each type is still maintained as part of the input to the RL algorithm, so as to maintain the same architecture for all cases, but in this case, the RL algorithm needs to learn that this information is useless. This case also required more training than the other scenarios we considered.

Fig. G.6 shows the performance boxplots for this scenario: in this case, performance is almost uniform, and all policies have a similar overall cost. The RL policy still has a small gain in terms of the overall cost, but it performs worse than the greedy Max policy on the max query. Performance on the count range query is almost uniformly good, and all differences between the policies are on the worst-case performance of the max query.

5.4 Mixed Query Arrival

Finally, we consider a third scenario: in this case, the max query follows a memoryless process with a probability 1/6 of generating a query at each time step, while count range queries are periodically generated every 6 slots. In this case, the policy needs to adapt to the possibility of a max query arriving, while also preparing for the foreseen count range queries.

The performance of the considered policies is shown in the form of boxplots in Fig G.7, as for the previous cases. The figure clearly shows that this case is much more

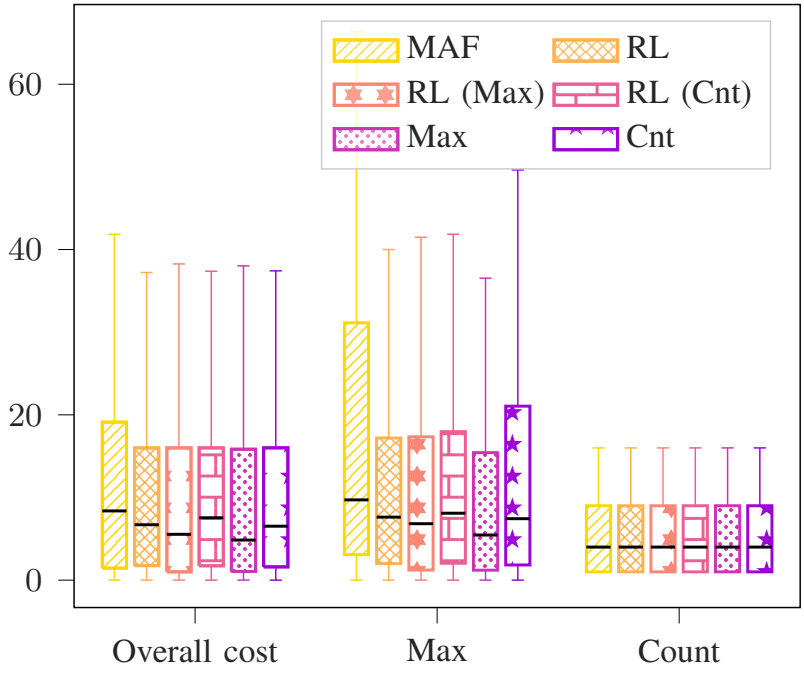


Fig. G.6: MSE cost of the different policies for both types of query in the geometric query scenario.

complex, and the RL policy does not manage to outperform the strategies that are oriented exclusively toward the max query. Since the max query is entirely unpredictable, the full RL policy would need more training to deal with this scenario: the simpler strategy learned by the RL (Max) scheme turns out to be better on average, while RL (Cnt) performs about as well as RL. In most cases, the error on the count range queries tends to be higher for all policies. We note, however, that the RL strategy still outperforms all others in terms of worst-case performance, as the 95th percentile whisker is particularly low for the count query, resulting in better overall worst-case performance. A better strategy could be learned with more training, and we note that the complexity of the scenario has a significant impact on the amount of training required, with mixed scenarios with deterministic and stochastic query processes being the most difficult.

By knowing the instants in which count range queries will arrive, the RL strategies can limit the worst-case error, although this comes at the cost of a slightly higher worst-case error on the max query (which is hard to optimize for, as its arrival process is completely unpredictable). In this case, as in the geometric query arrival scenario, the one-step greedy policy for the max query is actually performing almost as well as the RL version, as there is no long-term information to be learned on the query process. In

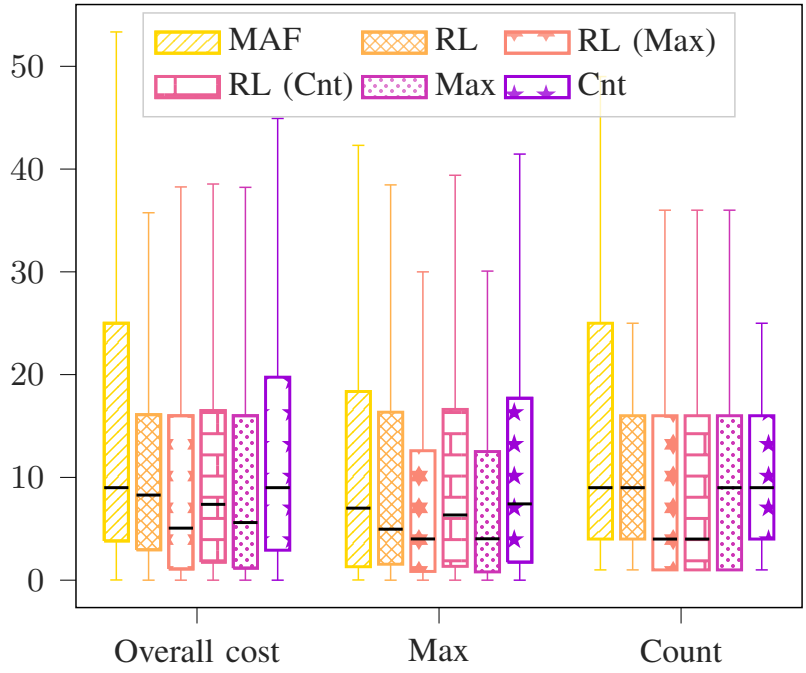


Fig. G.7: MSE cost of the different policies for both types of query in the mixed query scenario.

general, as was the case for QAoI, awareness of the query process is more useful if it is deterministic, or at least partially predictable.

6 Conclusions and Future Work

In this work, we have presented a framework for query-aware sensor scheduling, in which an edge node needs to choose the most relevant information to respond to external user queries, which may be different functions of the system state. This type of scenario is closely linked to semantic and task-oriented communications in the IoT, approaching the problem from a different angle: in our system, communications are pull-based, and the bottleneck of the system is medium access rather than rate, so that the solution is semantic, VoI-based scheduling rather than encoding.

Our work shows that query-aware scheduling can lead to profoundly different choices, depending on the specific functions that queries ask for and on the query arrival process for each client, and that RL-based strategies can provide a significant advantage in more predictable scenarios, while unpredictable query processes do not provide any

useful information to improve scheduling past one-step greedy strategies.

There are several open avenues of research to extend this work, both on the scheduling itself and on the process estimation. Firstly, scheduling is currently limited to a single sensors, and communication is entirely pull-based: a scenario in which multiple sensors can be polled at once, or sensors can transmit urgent information without being polled first, can make scheduling strategies more interesting. Furthermore, extending the problem from simple numeric values to richer types of information such as images or point clouds could prove useful to several applications, such as cooperative driving or robot swarm management, which require the integration of data-heavy information from multiple sources. This also leads to the second line of future work that we are exploring, i.e., the substitution of the Kalman filter with more complex estimators, such as deep networks, which can deal with much more complex functions and system models, and do not require prior knowledge of the system dynamics. Finally, the combination of a control system with the remote estimation would represent another step forward toward a fully task-oriented communication system.

References

[1] P. Popovski, O. Simeone, F. Boccardi, D. Gündüz, and O. Sahin, "Semantic-effectiveness filtering and control for post-5g wireless connectivity," *Journal of the Indian Institute of Science*, vol. 100, pp. 435–443, 2020.

[2] C. E. Shannon and W. Weaver, *The mathematical theory of communication.* University of Illinois Press, Sep. 1949.

[3] A. A. Soderlund and M. Kumar, "Optimization of multitarget tracking within a sensor network via information-guided clustering," *J. Guidance, Control, Dynamics*, vol. 42, no. 2, pp. 317–334, Feb. 2019.

[4] R. D. Yates, Y. Sun, D. Richard 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, Mar. 2021.

[5] P. Popovski, F. Chiariotti, K. Huang, A. E. Kalør, M. Kountouris, N. Pappas, and B. Soret, "A perspective on time toward wireless 6g," *Proceedings of the IEEE*, vol. 110, no. 8, pp. 1116–1146, 2022.

[6] F. Chiariotti, J. Holm, A. E. Kalør, B. Soret, S. K. Jensen, T. B. Pedersen, and P. Popovski, "Query age of information: Freshness in pull-based communication," *IEEE Trans. Comm.*, 2022.

[7] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled Software Defined Networking for efficient and scalable IoT communications," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 48–54, Sep. 2015.

[8] F. Chiariotti, A. E. Kalør, J. Holm, B. Soret, and P. Popovski, "Scheduling of sensor transmissions based on Value of Information for summary statistics," *IEEE Netw. Letters*, vol. 4, no. 2, pp. 92–96, May 2022.

[9] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Conf. on Computer Communications (INFOCOM)*. IEEE, Mar. 2012, pp. 2731–2735.

[10] Y. Sun and B. Cyr, "Sampling for data freshness optimization: Non-linear age functions," *Journal of Communications and Networks*, vol. 21, no. 3, pp. 204–219, 2019.

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

[12] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides, "The age of incorrect information: A new performance metric for status updates," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2215–2228, Jul. 2020.

[13] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 109–117.

[14] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," *IEEE/ACM Trans. on Networking*, vol. 28, no. 1, pp. 15–28, Dec. 2019.

[15] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, vol. 42, no. 2, pp. 251–260, Feb. 2006.

[16] A. Hashemi, M. Ghasemi, H. Vikalo, and U. Topcu, "Randomized greedy sensor selection: Leveraging weak submodularity," *IEEE Trans. Autom. Control*, vol. 66, no. 1, pp. 199–212, Mar. 2020.

[17] R. Duan, J. Du, J. Ren, C. Jiang, Y. Ren, and A. Benslimane, "VoI based information collection for AUV assisted underwater acoustic sensor networks," in *Proc. Int. Conf. Commun. (ICC)*. IEEE, Jun. 2020.

[18] S. M. Hoseyni, F. Di Maio, and E. Zio, "VoI-based optimal sensors positioning and the sub-modularity issue," in *Proc. Int. Conf. Syst. Rel. Safety (ICSRS)*. IEEE, Nov. 2019, pp. 148–152.

[19] G. Stamatakis, N. Pappas, A. Fragkiadakis, and A. Traganitis, "Autonomous maintenance in IoT networks via AoI-driven deep reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, may 2021.

[20] G. J. Stamatakis, N. Pappas, A. Fragkiadakis, and A. Traganitis, "Semantics-aware active fault detection in IoT," in *Proc. IEEE 20th Int. Symp. Model. Optim. Mobile, Ad hoc, Wireless Netw. (WiOpt)*, Sep. 2022.

[21] B. Yin, S. Zhang, Y. Cheng, L. X. Cai, Z. Jiang, S. Zhou, and Z. Niu, "Only those requested count: Proactive scheduling policies for minimizing effective age-of-information," in *Conf. on Computer Communications (INFOCOM)*. IEEE, Apr. 2019, pp. 109–117.

[22] M. E. Ildiz, S. Avşar, and E. Uysal, "An inequality for Query Age of Information and Age of Information," in *30th Signal Processing and Communications Applications Conference (SIU)*. IEEE, May 2022.

[23] M. E. Ildiz, O. T. Yavascan, E. Uysal, and O. T. Kartal, "Query Age of Information: Optimizing AoI at the right time," in *International Symposium on Information Theory (ISIT)*. IEEE, Jun. 2022, pp. 144–149.

[24] M. Hatami, M. Jahandideh, M. Leinonen, and M. Codreanu, "Age-aware status update control for energy harvesting IoT sensors via reinforcement learning," in *31st Ann. Int. Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, Aug. 2020.

[25] M. Hatami, M. Leinonen, and M. Codreanu, "AoI minimization in status update control with energy harvesting sensors," *IEEE Trans. on Communications*, Sep. 2021.

[26] C. Xu, X. Wang, H. H. Yang, H. Sun, and T. Q. Quek, "AoI and energy consumption oriented dynamic status updating in caching enabled IoT networks," in *Conf. on Computer Communications Worksh. (INFOCOM)*. IEEE, Jul. 2020, pp. 710–715.

[27] X. Zheng, S. Zhou, and Z. Niu, "Urgency of information for context-aware timely status updates in remote control systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7237–7250, 2020.

[28] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *Proc. ACM Int. Conf. Auton. Agents Multi-Agent Syst. (AAMAS)*, May 2013, pp. 1101–1108.

[29] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NeurIPS*, Dec. 2016, pp. 2145–2153.

[30] T.-Y. Tung, S. Kobus, J. P. Roig, and D. Gündüz, "Effective communications: A joint learning and communication framework for multi-agent reinforcement learning over noisy channels," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2590–2603, Jun. 2021.

[31] A. OroojlooyJadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *arXiv preprint arXiv:1908.03963*, Aug. 2019.

[32] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960.

[33] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE Trans. Automat. Contr.*, vol. 49, no. 9, pp. 1453–1464, 2004.

[34] A. M. Mathai and S. B. Provost, *Quadratic forms in random variables: theory and applications*. Dekker, 1992.

[35] F. Amram, "Multivariate extreme value distributions for stationary Gaussian sequences," *Journal of multivariate analysis*, vol. 16, no. 2, pp. 237–240, Apr. 1985.

[36] D. Luengo, L. Martino, M. Bugallo, V. Elvira, and S. Särkkä, "A survey of Monte Carlo methods for parameter estimation," *EURASIP J. Adv. Signal Process.*, vol. 2020, pp. 1–62, Dec. 2020.

[37] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[38] R. A. Howard, *Dynamic programming and Markov processes*. John Wiley, 1960.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[40] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[41] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-735-2

AALBORG UNIVERSITY PRESS