AN ABSTRACT OF THE DISSERTATION OF

Ethan W. Dereszynski for the degree of Doctor of Philosophy in Computer Science presented on June 4, 2012.

Title: Probabilistic Models for Quality Control in Environmental Sensor Networks

Abstract approved: _____

Thomas G. Dietterich

Networks of distributed, remote sensors are providing ecological scientists with a view of our environment that is unprecedented in detail. However, these networks are subject to harsh conditions, which lead to malfunctions in individual sensors and failures in network communications. This behavior manifests as corrupt or missing measurements in the data. Consequently, before the data can be used in ecological models, future experiments, or even policy decisions, it must be quality controlled (QC'd) to flag affected measurements and impute corrected values. This dissertation describes a probabilistic modeling approach for real-time automated QC that exploits the spatial and temporal correlations in the data to distinguish sensor failures from valid observations. The model adapts to a site by learning a Bayesian network structure that captures spatial relationships among sensors, and then extends this structure to a dynamic Bayesian network to incorporate temporal correlations. The final QC model contains both discrete and continuous variables, which makes inference intractable for large sensor networks. Consequently, we examine the performance of three approximate methods for inference in this probabilistic framework. Two of these algorithms represent contemporary approaches to inference in hybrid models, while the third is a greedy search-based method of our own design. We demonstrate the results of these algorithms on synthetic datasets and real environmental sensor data gathered from an ecological sensor network located in western Oregon. Our results suggest that we can improve performance over networks with less sensors that use exhaustive asynchronic inference by including additional sensors and applying approximate algorithms.

©Copyright by Ethan W. Dereszynski
June 4, 2012
All Rights Reserved

Probabilistic Models for Quality Control in Environmental Sensor Networks

by

Ethan W. Dereszynski

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 4, 2012
Commencement June 2013

Doctor of Philosophy dissertation of Ethan W. Dereszynski presented on June 4, 2012.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Ethan W. Dereszynski, Author

ACKNOWLEDGEMENTS

The author would like to thank the graduate committee for their advice and guidance, in addition the Ecosystem Informatics IGERT Program and its PIs for their financial and academic support.

This research was partly funded by ARO grant W911NF-08-1-0242. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO or the United States Government.

This research was partially supported by the Defense Advanced Research Projects Agency under DARPA grant FA8650-06-C-7605. Views and conclusions contained in this document are those of the author and do not necessarily represent the official opinion or policies, either expressed or implied of the US government or of DARPA.

This material is based upon work supported by the National Science Foundation under Grant Nos. 0832804, 0333257, and 0307592. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

CONTRIBUTION OF AUTHORS

The author would like to thank Frederick Bierlmaier, Donald Henshaw, and Suzanne Remillard for providing us with the raw and processed atmospheric data from the H.J. Andrews LTER and for their help in discerning the anomaly types found therein. We thank Marc Parlange and the EFLUM Group for hosting and supporting part of this research at the EPFL. We would also like thank Mounir Krichane and Guillermo Barrenetxea for providing us with the SensorScope datasets and for their technical advice regarding the sensor stations.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

Figure Page

LIST OF TABLES

LIST OF ALGORITHMS

This work is dedicated to my wife, Elizabeth. Her unwavering love, encouragement, and support convinced me that I would achieve this conclusion. I am happy to say that, once again, she was right. If I am a better man at the end, it is for the grace that she has shown me. This work is also dedicated to my mother, Deborah. When I was a child you told me that I could be anything I ever wanted to be. And as I grew up, you gave everything you had to help me achieve my dream. I love you both. Thank you.

Chapter 1: Introduction

1.1 Introduction

Automated sensor technology has revolutionized the way ecosystem scientists acquire knowledge about our environment. Scientists are no longer limited to manually collecting observations at accessible field sites and sampling at limited or irregular time intervals. A single in-situ sensor can record measurements from isolated locations at high temporal resolution, and relay these sensor readings to a data repository for analysis and distribution to the broader community. Moreover, advances in sensor technology have decreased costs and improved availability. As a result, ecologists can now deploy dense networks of sensors to capture environmental processes at increasingly finer spatial resolutions. The final product is an overall picture of the landscape, along with the complex ecological processes at work within it, at a level of detail previously unachievable through conventional sampling techniques.

Ecological research organizations, such as the Long Term Ecological Research (LTER) network and National Ecological Observatory Network (NEON), have adopted these technologies to create continental-scale sensor networks. A goal of these agencies is to identify the drivers behind, and validate existing models of, ecological processes occurring at different spatiotemporal scales using measurements gathered from instrumented sites. However, before such a task can be undertaken, sensor data must be quality controlled (QC'd) to remove invalid readings caused by sensor failure. This is particularly relevant to environmental sensor data, as the in-situ nature of the sensors makes them prone to malfunction. Such malfunctions are exhibited by biased readings, calibration errors, signal loss, and additional failures not yet known. As model forecasts and analyses of the collected data are the primary inputs to data-driven ecological research, and ultimately influence policy decisions, it is paramount that the data undergo QC prior to its integration.

In this dissertation, we propose a set of algorithms to automate quality control of environmental sensor data. Data from a sensor network is treated as a multivariate time series, with each dimension of the series corresponding to measurements taken by a single sensor at a fixed sampling frequency. Given such a dataset, an ideal quality control scheme should demonstrate three qualities:

- The approach must flag observations that are likely to be corrupted by sensor failure.

- It should support a means of gap filling by providing a best-guess imputation of the affected observation.
- The scheme should be generalizable to different types of sensor data, deployment locations, and fault types.

We refer to a measurement corrupted by sensor failure as a "data anomaly" so as not to confuse it with the more general "anomaly," which may refer to an unusual measurement caused by real environmental events. Related to the first two properties, a confidence measure in the classification of an observation being "good" or a "data anomaly," as well as the imputation of affected values, should also be provided. The confidence measure allows consumers of the data to determine what values are acceptable for their purposes.

Many of the approaches used by information managers to perform QC typically fall short in one of the aforementioned objectives. For example, a common method is to apply a series of range checks and remove measurements that fall outside of acceptable levels (for example, air temperature measurements > 50 ℃). While fast to apply to large volumes of data, these methods fail to detect data anomalies that may occur within reasonable limits, nor can they distinguish values on the edge of these limits (questionable measurements) from those in the middle (nominal measurements) [64]. After initial range checks, a common follow-up is manual inspection of the plotted time series by a domain expert. Experts can identify abnormal behavior of a sensor based on their knowledge of the hardware and the phenomenon it is measuring. Unfortunately, such inspection is infeasible as the number of sensors grows large and each sensor records at very fine temporal resolutions. Neither range checks nor visual inspection provides a solution for gap-filling in place of affected values.

The work described in this dissertation focuses on a machine learning approach, wherein we learn a probabilistic model $P(\mathbf{X}^t|\mathbf{X}^{t-1}, \mathbf{X}^{t-2}, \ldots)$ of the latent process generating observations at each sensor. This process model is learned directly from the sensor data. Uncertainty in the working state of the sensor is encoded in a probability distribution over its state $P(S)$. A sensor model $P(O|X, S)$ couples sensor observations O to the process model according to the state of the sensor. In effect, we treat sensor measurements as noisy observations of this latent process, where the degree of noise is linked to the functioning state of the sensor; i.e., whether the sensor is *working* or *broken*. Queries regarding the working state of the sensor (and the quality of its readings) are

resolved through statistical inference. An advantage of the probabilistic framework is that it provides a natural interpretation of "confidence" in the form of probability values. The data-driven learning of the process model minimizes the amount of domain expert knowledge required to configure this method to a new site. Moreover, by avoiding modeling specific ways in which a sensor can fail, the model can generalize to new malfunctions that may arise over the course of a deployment.

Before introducing the contributions in this dissertation, we note that this work extends research completed as part of the author's Masters thesis [16]. There, we applied a dynamic Bayesian network (DBN) model that incorporated a learned baseline component to make predictions about future observations for a single environmental sensor. The baseline function $b(qh, d)$ analyzed a historical record of observations from the sensor to calculate a smoothed estimate of the air temperature reading at a given quarter-hour qh and day d. To learn the baseline, we adopted techniques from time series analysis to isolate trend effects caused by diurnal and seasonal cycles present in the data. Once those effects were accounted for by the baseline, the residuals were modeled using a first-order Markov process. The behavior of the (Gaussian) random walk accounted for to temporally-localized weather events (warming periods, cooling periods, storm events, etc.).

1.2 Contributions

In this section, we summarize our contributions to quality control for environmental sensor data.

Our first contribution overcomes two shortcomings of the aforementioned single-sensor model. First, the single-sensor model only considers one sensor's observations when inferring the value of the latent process. If the sensor state becomes *broken*, our uncertainty about the value of the latent process will grow larger each time step, because the model will disconnect the observation from updating its belief about the process. Consequently, we will have no way to impute what the sensor "should" have observed beyond the baseline value. Second, there exist many cases where we cannot learn the baseline function. This is because training the baseline requires an existing archive of past observations, which must contain at least one valid sensor reading from every time step in the period of the seasonal and diurnal trends. If such a record does not exist, as in the case of short-

term deployments, we cannot establish a profile of regular trend effects for the sensor. Here, "short-term" refers to a deployment with a duration shorter than a full cycle of the process(es) it is monitoring (for example, air temperature sensors deployed for less than a full year).

In Chapter 2, we introduce a solution that addresses both of these shortcomings by incorporating multiple sensors at a site. Specifically, we learn a linear-Gaussian Bayesian network that encodes a joint distribution $P(\mathbf{X}) = P(X_1^{t-1}, X_1^t, \ldots, X_N^{t-1}, X_N^t)$ over a *set* of processes being tracked by multiple environmental sensors in a network[18]. The joint distribution over the process includes two components. The first is a temporal component $P(\mathbf{X}^t|\mathbf{X}^{t-1})$ that specifies a transition function to capture how the set of processes evolves from time $t-1$ to time t. A second component $P(X_i^t|\mathbf{X_j}^t \subset \mathbf{X}^t)$ represents the set of asynchronic (within a time slice) dependencies, and reflects the spatial relationship among the sensors. Each variable in $P(\mathbf{X})$ is represented by a node in a directed acyclic graph (DAG), and edges in the graph indicate conditional dependencies among the variables. The structure of the linear-Gaussian network is learned via a hill-climbing search algorithm and the BGe scoring metric [24]. A result of our approach was that the learned process models often generalized better to new sensor observations than models that ignored either the spatial or temporal dependencies, or models that assumed full connectivity in the structure of $P(\mathbf{X})$. The ability of the model to generalize to new test data was noted by its superior performance in the QC task, in addition to other metrics described in the manuscript.

While this approach performed well on data gathered from short-term deployments, it highlighted a scaling issue associated with inference in our probabilistic QC model. Specifically, our method for inference in the multisensor model required computational time that grew exponentially in the number of sensors N. This ultimately limited our analysis to networks containing fewer than 10 sensors. In order to scale to modern sensor networks, which may consist of dozens to hundreds of sensors, we needed to explore approximate methods for performing inference in our model.

The second significant contribution of this dissertation addresses the problem of scale by considering three approximate algorithms for inference in our QC model. In Chapter 3, we describe the methods of Rao-Blackwellized particle filtering (RBPF), Expectation Propagation (EP), and SearchMAP, our own greedy-based method for approximate inference, as applied to our probabilistic QC model. Further, we evaluate how the inclusion

of additional sensors (and sensor types) affects the overall performance of the QC model. In particular, we examine the tradeoff between inaccuracies introduced by approximate algorithms for asynchronic inference versus the increased precision of a larger process model that incorporates observations from additional sensors. As a case study, we apply these algorithms to a network of 27 sensors from the H.J. Andrews Experimental Forest. The algorithms were evaluated using both raw data and injected synthetic noise to simulate some of the more common types of data anomalies. Results in that work suggest that the benefits of including more sensors in the QC model justify an approximate approach to inference; however, we noted that diminishing returns in performance arose after a number of sensors had been added to the model. A surprising finding was that our greedy search method for determining the maximum aposteriori (MAP) assignment of sensor-state variables outperformed both EP and RBPF in detecting data anomalies in the data and imputing the affected values.

By mitigating the high computation cost of additional sensors, we have opened our research to several new challenges. These challenges include finding more accurate representations for the correlations among sensors in a network, transferring knowledge from learned models of the process to new deployments, and developing methods for learning disjoint components of the process model. We conclude with a discussion of these challenges in Chapter 4. A summary our contributions to quality control of ecological sensor data is as follows:

1. We established an approach that leverages correlation among a network of environmental sensors to perform real-time, simultaneous quality control for multiple sensors.
    (a) The approach included a Bayesian methodology for learning spatiotemporal structure among correlated ecological sensors.
    (b) Our results featured a case study that demonstrates the application of the QC model to short-term deployments.
2. We performed an analysis of approximate inference methods for asynchronic inference in probabilistic QC models.
    (a) We included a treatment of implementation issues regarding advanced algorithms for inference in hybrid dynamic Bayesian networks.

(b) A greedy-search method for approximate inference was introduced for the QC domain.

(c) The experiments assessed the benefits gained from incorporating additional sensors into a QC model versus incurred costs of approximate inference.

Chapter 2: Manuscript One

Spatiotemporal Models for Data-Anomaly Detection in Dynamic Environmental Monitoring Campaigns

Ethan W. Dereszynski and Thomas G. Dietterich

ACM Transactions on Sensor Networks
New York, NY
Vol. 8, No. 1, August 2011
Pages 3:1–3:36

2.1 Abstract

The ecological sciences have benefited greatly from recent advances in wireless sensor technologies. These technologies allow researchers to deploy networks of automated sensors, which can monitor a landscape at very fine temporal and spatial scales. However, these networks are subject to harsh conditions, which lead to malfunctions in individual sensors and failures in network communications. The resulting data streams often exhibit incorrect data measurements and missing values. Identifying and correcting these is time-consuming and error-prone. We present a method for real-time automated data quality control (QC) that exploits the spatial and temporal correlations in the data to distinguish sensor failures from valid observations. The model adapts to each deployment site by learning a Bayesian network structure that captures spatial relationships between sensors, and it extends the structure to a dynamic Bayesian network to incorporate temporal correlations. This model is able to flag faulty observations and predict the true values of the missing or corrupt readings. The performance of the model is evaluated on data collected by the SensorScope Project. The results show that the spatiotemporal model demonstrates clear advantages over models that include only temporal or only spatial correlations, and that the model is capable of accurately imputing corrupted values.

2.2 Introduction

The increasing availability (coupled with decreased cost) of lightweight, automated wireless sensor technologies is changing the way ecosystem scientists collect and distribute data. Portable sensor stations allow field experts to transport monitoring equipment to sites of interest and observe ecological phenomena at a spatial granularity of their choosing. These nonpermanent deployments stand in stark contrast to traditional observatory-like environmental monitoring stations whose initial spatial layout remains unchanged over the course of time. However, both approaches are providing researchers with an unprecedented volume of ecological data. The resultant surge in data has potential to transform ecology from an analytical and computational science into a data exploration science [66].

Temporary sensor deployments, whose durations can range from a single week to several months, represent a new challenge for data quality control. By nature of being

in-situ environmental stations, they are prone to the same technical problems as long-term deployments, namely damage due to extreme weather, transmission errors and loss of signal, calibration errors, and drastic changes in environmental conditions. Further, it is important to rapidly detect and diagnose a damaged or failing sensor so that it can be repaired. An insufficiently fast diagnosis could result in the corruption or loss of data from a given sensor for the duration of the deployment; consequently, techniques involving a postmortem analysis of the data are of little or no value. However, the sheer abundance of data provided by large sensor networks operating at fine time resolutions makes manual analysis (visualizing the data) infeasible for both online and offline quality control. This raises the need for efficient automated methods of data "cleaning" that can function in an online setting and that can readily adapt to dynamic spatial distributions.

The purpose of this article is to provide an example of spatially distributed environmental monitoring, motivate a need for quality control in this domain through documented examples of sensor failure, and introduce a machine learning approach to automate the data cleaning process. Though we believe our methodology is readily extendable to additional environmental phenomena, our work here deals only with air temperature data. We propose an adaptive quality control (QC) system that exploits both temporal and spatial relationships among multiple environmental sensors at a site. The QC system makes use of a dynamic Bayesian network (DBN, [14]) to correlate sensor readings within a sampling period (time step) to readings taken from past sampling periods. Because the set of potential faults is unbounded, it is not practical to approach this as a diagnosis problem where each fault is modeled separately [31]. Instead, we employ a general fault model and focus on creating a highly accurate model of normal behavior, known as the process model. The intuition is that if there is a discrepancy between the current estimate of normal behavior (provided by the process model) and the observation taken from the sensor, then the observation is labeled as anomalous. An additional benefit of this approach is that it can impute values for the sensor readings during periods of sensor malfunction.

This article is organized as follows. First, we will discuss the current ecological monitoring campaign, known as SensorScope, that produced the data studied herein. Second, we describe the nature of the air temperature data and the data-anomaly types encountered, followed by a introduction to hybrid Bayesian networks. Third, we describe our quality control model, including learning the process model and incorporating a general

fault model. Finally, we present the results of the model applied to temperature data from select SensorScope deployments as well as empirical results on synthetic data. We conclude with a plan for future research.

2.3 The SensorScope System

The SensorScope Station, developed at the École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland, represents a significant change in traditional tools for in-situ data collection and distribution. In place of few, expensive long-term or permanent monitoring stations deployed sparsely over a heterogeneous spatial area, SensorScope allows field scientists to deploy many light weight, inexpensive stations at a much higher spatial resolution and monitor at user-specified time granularities. The portability of these stations facilitates dynamic deployments, wherein sensors can be relocated within a deployment to adapt to changing monitoring requirements.

Component sensors for measuring air temperature, skin (surface) temperature, wind speed, wind direction, humidity, etc. are typically acquired from external manufacturers. The SensorScope stations are equipped with a power supply sufficient to host a small set of these sensors (the number dependent on each sensor's energy requirement) operating simultaneously, as well as a radio device for communication with nearby stations. Every deployment contains at least one General Packet Radio Service (GPRS) hub that transmits data received from the SensorScope stations to a central server at the EPFL via cellular signal. Once the data reaches the central server, it is converted from its raw voltage to a value particular to the phenomenon being measured (degrees Celsius in the case of temperatures) via a conversion formula specific to the sensor type. When the data is requested for download or plotting via the SensorScope Web site,[1] it is filtered automatically by a range checker to remove extreme values associated with sensor malfunctions.

Figure 2.1 (left) shows a 3-D visualization of the Le Genepi Glacier deployment, which was in place from August 27 to November 5 of 2007. The glacial valley, located approximately 60 kilometers south of the western edge of Lake Geneva, slopes downward toward the northeast and is surrounded by mountains on all other sides. The sensors are placed at an elevation range of 2300 meters to 2500 meters. At the time of the deployment,

[1] `http://sensorscope.epfl.ch/index.php/SensorScope_Deployments`

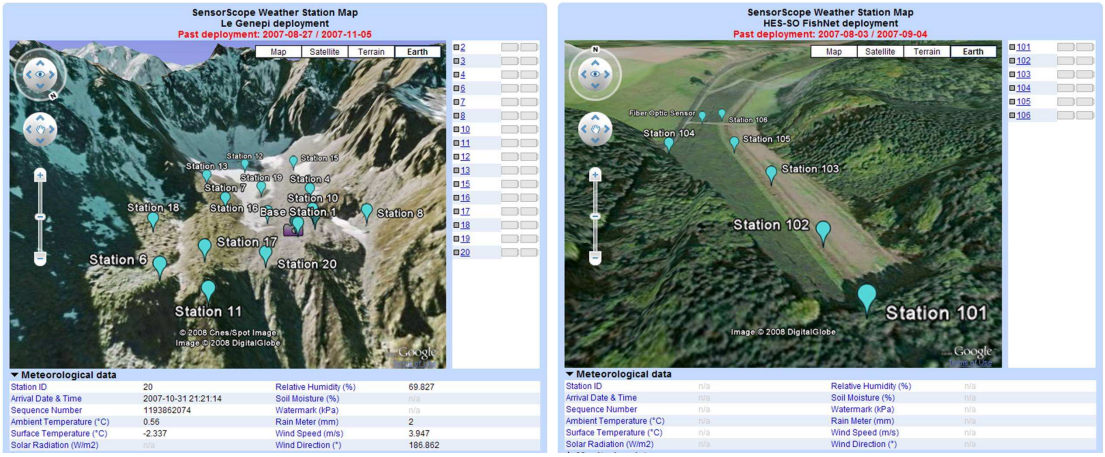

Figure 2.1: The Genepi Glacier and FishNet SensorScope deployments (from Google Earth)

the northeast corner of the glacier was the only area accessible by cellular signal; therefore, the GPRS (labeled "Base Station 1") was placed in this location. A total of 16 stations were deployed over the area, whose dimensions can be roughly approximated by a 100 meter by 200 meter rectangle, to allow for a comprehensive analysis incorporating the spatial heterogeneity of the relatively small region.

The right portion of Figure 2.1 shows a much smaller deployment of six stations along a stream, known as the FishNet deployment. The sensors operated from from August 3 to September 4 of 2007. The topographical difference is relatively small compared to Le Genepi (the sensors are all located at approximately 600 meters of elevation), as the deployment was in an agricultural area bordering a forested area to the south. The GPRS station (not shown in the figure) is located approximately 100 meters to the west of Station 104, and the length of stream covered by the deployment is approximately 300 meters.

2.4 SensorScope Data

Each SensorScope station is capable of hosting a changing set of environmental sensors; hence, there is not a consistent set of phenomena recorded by all stations across all deployments. Rather, each station is provided only with those sensors needed to measure

the variables of interest for a given field campaign and is then retooled between deployments. As air temperature (the temperature roughly 1.5 meters above the surface) is of interest in nearly all campaigns to date, we shall focus our discussion primarily on this type of data. Air temperature readings are taken from Sensirion SHT75 sensors mounted on the SensorScope stations [63]. Figures 2.2 and 2.3 show two different sets of data streams from the stations at the FishNet and Le Genepi deployments, respectively. The air temperature readings from both sites were sampled at a rate of once every two minutes. The graphs show those readings binned and averaged into 10-minute windows.

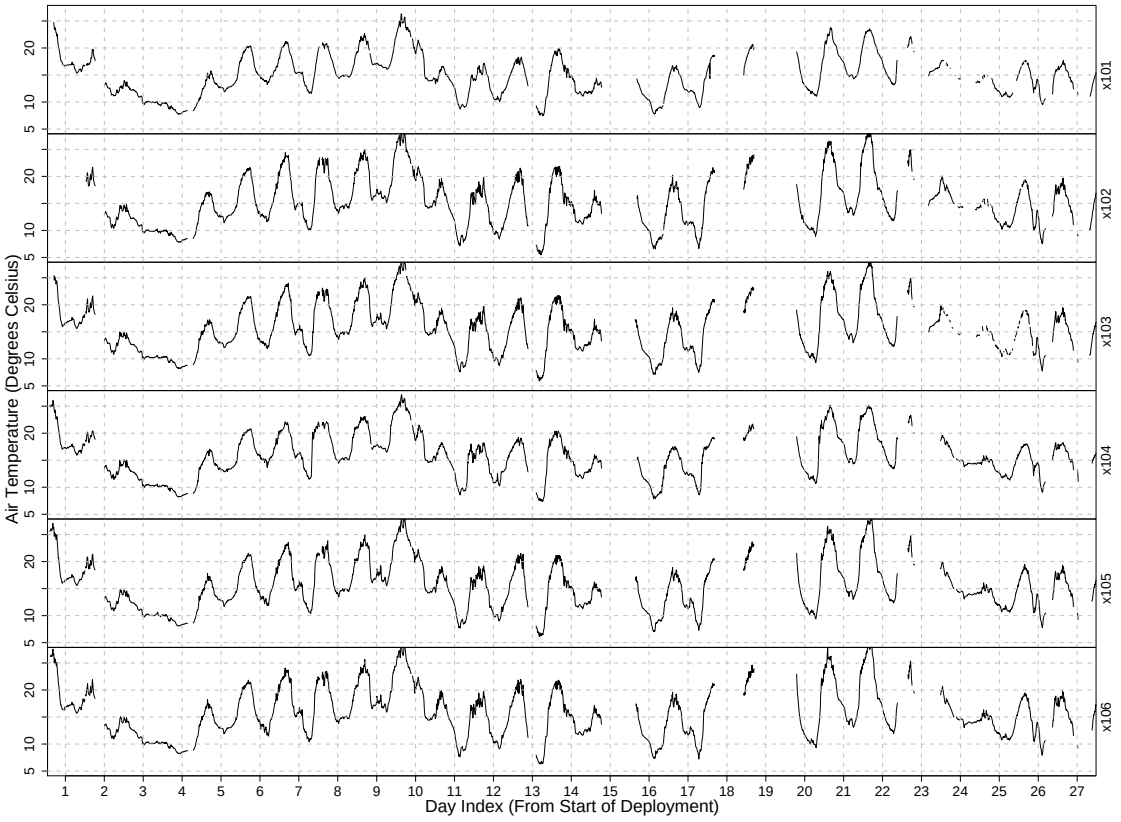

Figure 2.2: Air temperature readings from the FishNet SensorScope deployment. Each row represents the sensor labeled in the corresponding upper right corner. The X axis denotes the day (vertical dashed line depicts midnight) since the deployment began, and the Y axis denotes temperature in degrees ℃. Corresponding station names appear on the right side of the graph next to the stream that they depict.

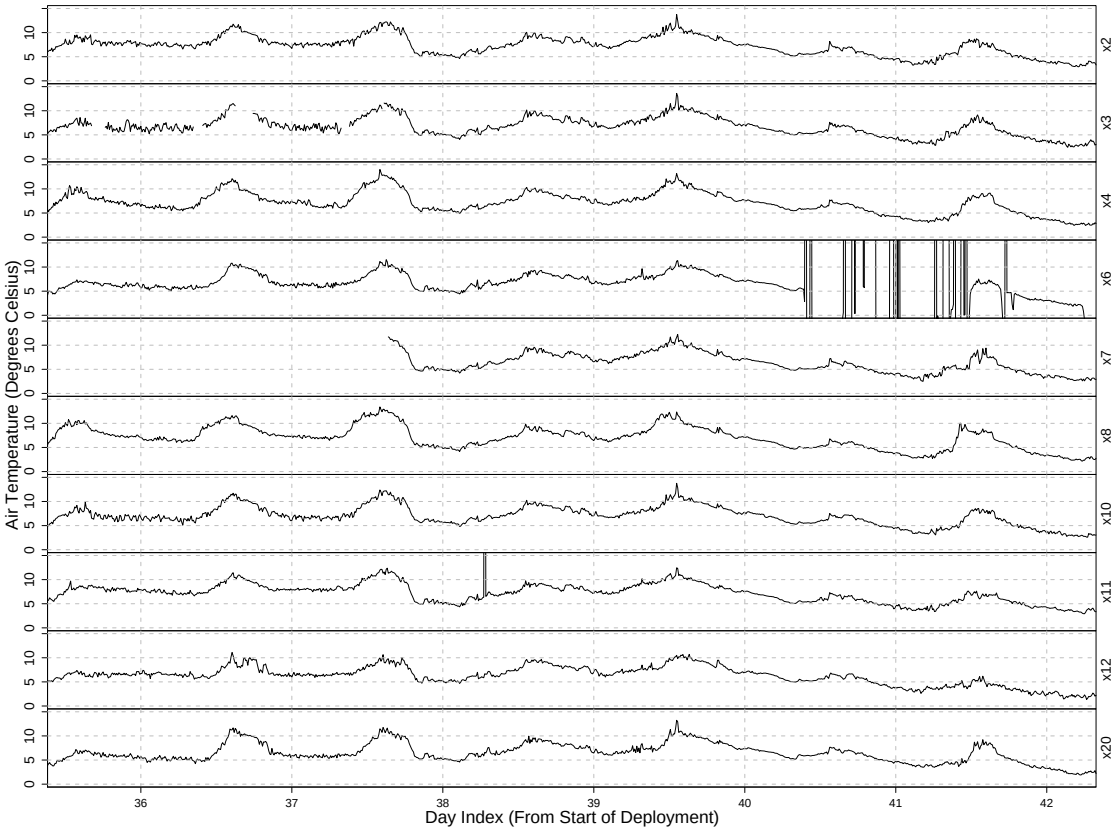

Figure 2.3: Air temperature readings from one week at the Le Genepi deployment. Each row represents the sensor labeled in the corresponding upper right corner. The X axis denotes the day (vertical dashed line depicts midnight) since the deployment began, and the Y axis denotes temperature in degrees ℃. Corresponding station names appear on the right side of the graph next to the stream they depict.

Nominal air temperature data contains a regular diurnal (day to day) trend that is dependent on the season and location of the sensor. For example, the FishNet has a more pronounced diurnal signal because the recording period is in the late summer (August) whereas the the Le Genepi deployment has a suppressed diurnal trend due to both the time it was observed (October) and its Alpine location. Storm and cloud coverage events occur at irregular intervals but may also suppress diurnal signal. The FishNet data shows the effect of a storm in days 2 through 4. We have found the following data-anomaly types present in the air temperature data:

- *GPRS Outage.* In the case where the GRPS hub becomes inoperative, data for the entire deployment is lost. The FishNet deployment contains many segmented periods of sensor outages among all stations. These outages indicate a failure of the GPRS, because they occur simultaneously across all sensor streams. The faults are evident between days 15-16 and days 17-20.

- *Sensor Outage.* A sensor outage occurs when an individual sensor stream is lost. Multiple sensor outages can overlap during a give time period; however, unlike in a GPRS outage, the start, end, and duration of each outage is not synchronized. There are individual sensor outages in the FishNet deployment at stations 101, 102, and 103, spanning days 24 and 25.

- *Data Anomalies.* Data anomalies are characterized as observations from a given sensor that are corrupted due to sensor malfunction. Such anomalous values are particularly obvious at station 6 in the Le Genepi deployment (Figure 2.3), where extremely large temperature values are recorded due to incorrect voltages generated at the sensor. Subtler spikes in temperature occur at station 6 on day 41 (Le Genepi) and station 101 on day 17 (FishNet). A flatline in temperature is created by the temperature sensor reporting a 0-voltage value. The conversion algorithm maps this value to -1 ℃ value upon storing it into a data base. An example of this error is provided in Section 2.7.2.

Given the correlation between the sensors within a deployment, it is our goal to be able to identify data anomalies and impute the true temperature values in the case of both individual sensor outages and sensor malfunctions. Sensor failures that manifest themselves as either extraordinarily hot or cold temperatures are simple to diagnose by means of range checking [49]; however, malfunctions resulting in flatline values and subtler spikes in temperature readings are not detected by extreme value tests. While the values appear anomalous in the context of their immediate temporal neighbors, they are not abnormal in the range of temperatures recorded over the full duration of the deployment.

2.5 Hybrid Bayesian Networks

Our probabilistic model of the air temperature domain is a conditional linear-Gaussian network, also known as a hybrid network because it contains both continuous and discrete variables [40, 51, 54]. For the sake of computational convenience, we will restrict our networks so that discrete-valued variables do not have continuous-valued parents and so that all continuous-valued variables are modeled as Gaussians.

In this section, we describe how the probability distributions for continuous-valued variables are parameterized. We consider three cases: (a) continuous variables with discrete parents, (b) continuous variables with continuous parents, and (c) continuous variables with a mix of discrete and continuous parents.

2.5.1 Continuous Variables with Discrete Parents

Consider a single continuous variable, X. For every possible instantiation of values for the discrete parents of X, X takes on a Gaussian distribution with separate values for μ and σ^2. For example, if X has a single Boolean parent, $Y = y \in \{true, false\}$, then the conditional probability table (CPT) of X would contain two entries: $P(X|y = true) \sim N(\mu_t, \sigma_t^2)$ and $P(X|y = false) \sim N\left(\mu_f, \sigma_f^2\right)$. In general, let $\mathbf{Y} = \{Y_1, Y_2, ..., Y_n\}$ denote the set of discrete parents of the continuous variable X. Further, let $|\mathbf{Y}| = |Y_1| \times |Y_2| \times ... \times |Y_n|$ be the total size (number of possible instantiations) of $\mathbf{Y}$. Then, we specify the CPT of X with the $|\mathbf{Y}|$ dimensional vector, $\vec{\mu} = \langle \mu_1, \mu_2, ..., \mu_{|\mathbf{Y}|} \rangle$. Similarly, we specify the set of variances of X, depending on the parent configuration, as the vector $\vec{\sigma^2} = \left\langle \sigma_1^2, \sigma_2^2, ..., \sigma_{|\mathbf{Y}|}^2 \right\rangle$. Figure 2.4 (left) contains an example with binary discrete variables.

2.5.2 Continuous Variables with Continuous Parents

Consider a single continuous variable, X, but now with m continuous-valued parents. For each continuous-valued parent, $Z_i \in \mathbf{Z} = \{Z_1, Z_2, ..., Z_m\}$, X has a weight, w_i, such that mean of X is calculated as:

$$\mu_x \quad = \quad \epsilon + \sum_{i=1}^{m} w_i z_i, \tag{2.1}$$

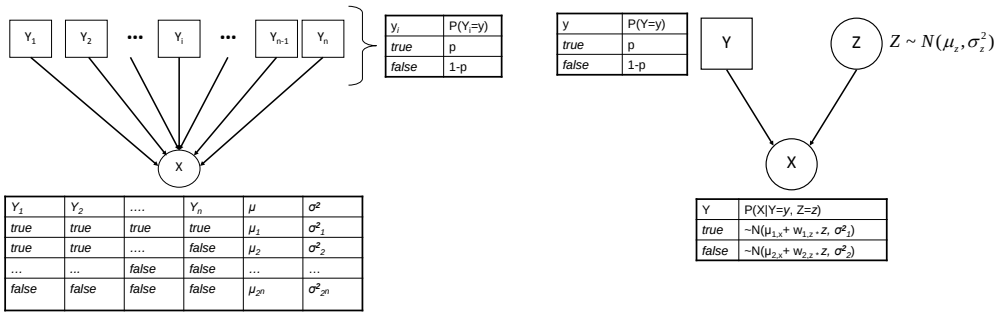

Figure 2.4: Left: Conditional Gaussian Bayesian Network. Right: Conditional Linear-Gaussian Bayesian Network

where z_i is the value of the parent random variable, Z_i, and ϵ is X's "intercept term" in the linear regression formula. An essential requirement for computational tractability is that the variance of X is specified by a single σ^2 parameter and which is not conditioned on the parents. Note that this conditional distribution has exactly the same form as a linear regression model.

2.5.3 Continuous Variables with a Mix of Discrete and Continuous Parents

If a variable has a mix of continuous and discrete parents, we employ a distinct linear Gaussian distribution for each combination of values of the discrete parents. This is known as a Conditional Linear Gaussian (CLG) model. Let X be a continuous variable with a set of discrete parents, $\mathbf{Y}$, and continuous parents, $\mathbf{Z}$. Then X has a separate mean, variance, and set of regression weights for each possible instantiation of $\mathbf{Y}$. We specify a CLG variable by a mean vector, $\vec{\mu} = \left\langle \mu_1, \mu_2, ..., \mu_{|\mathbf{Y}|} \right\rangle$, a variance vector, $\vec{\sigma^2} = \left\langle \sigma_1^2, \sigma_2^2, ..., \sigma_{|\mathbf{Y}|}^2 \right\rangle$, and a $|\mathbf{Y}| \times |\mathbf{Z}|$ regression matrix:

$$\begin{bmatrix} w_{1,1} & w_{1,2} & ... & w_{1,|\mathbf{Z}|} \\ w_{2,1} & w_{2,2} & ... & w_{2,|\mathbf{Z}|} \\ w_{...,1} & w_{...,2} & ... & w_{...,|\mathbf{Z}|} \\ w_{|\mathbf{Y}|,1} & w_{|\mathbf{Y}|,2} & ... & w_{|\mathbf{Y}|,|\mathbf{Z}|} \end{bmatrix}. \tag{2.2}$$

Figure 2.4 (right) contains a CLG network with a continuous variable having one discrete and one continuous parent.

2.6 Developing a Spatiotemporal Process Model

In the following sections, we describe our process model as the combination of two individual pieces: the spatial component that represents the relationships among all sensors within a deployment for a given time slice, and a temporal component that captures the transition dynamics from one time period (10-minute interval) to the next. The complete process model is represented as a Dynamic Bayesian Network [14], in which the task of quality control is achieved by inferring the most likely state of the sensors given the current temperature observations and those of the immediate past. The procedure for building the complete quality control model is summarized in Table 2.6.4.

2.6.1 Structure Learning and the Spatial Component

As the geographical layout of stations changes with each deployment, it is desirable to have the QA routines autonomously learn the spatial relationships for each new deployment from the observed data. To this end, we apply Bayesian network structure learning algorithms to learn the set of conditional-independence relationships among sensors at a given deployment. Recall that though each SensorScope station is capable of monitoring several environmental variables according to the type and number of sensors it is hosting, we focus our work to air temperature data for purposes of site-to-site continuity and ignore other data types. Further, we assume that each set of air temperature observations corresponding to a single 10-minute period is generated from a multivariate Gaussian distribution, and thus a sample from a deployment containing n SensorScope stations is generated from an n-dimensional multivariate Gaussian. Our goal then is to learn the elements of the covariance matrix that determine how each dimension (each sensor in a deployment) relates to the others. Our assumptions facilitate the application of a measure known as BGe (Bayesian metric for Gaussian networks having score equivalence, developed by Geiger and Heckerman [24]) as a scoring metric for candidate networks. We summarize the scoring metric, but ask the interested reader to see the aforementioned reference for further details.

A Bayesian network over n Gaussian distributed variables has a joint distribution equal to a n-dimensional multivariate Gaussian. The network structure is referred to as a sparse representation of the multivariate distribution. "Sparse" in this context means that the representative network may not directly correlate each variable with all other variables; in graphical terms, less than $\frac{n(n-1)}{2}$ edges (the maximum amount of edges for an acyclic graph) are sufficient to represent the covariance structure among n variables. In general, a lesser degree of connectivity in the graph structure will decrease the time required to perform inference in the model and reduce the number of parameters to fit once the structure is determined.

The BGe metric assumes the existence of a prior linear Gaussian network whose full joint distribution represents an initial estimate of the true distribution from which the observations are drawn. This Bayesian network can either be knowledge-engineered by domain experts with information about the deployment or constructed ad hoc in cases where specific domain knowledge is absent. Geiger and Heckerman parameterize the unknown generative, multivariate distribution with a mean vector, $\vec{m}$, and precision matrix, $W = \Sigma^{-1}$. The prior joint distribution on these parameters is assumed to be a normal-Wishart. Under these assumptions, the joint posterior distribution, given a data set D (containing multivariate observations $\vec{x}_1, \vec{x}_2, ..., \vec{x}_l$, each of n dimensions), over $\vec{m}$ and W can be divided into the conditional distribution $P(\vec{m}|W)$ and the marginal $P(W)$. The conditional distribution of $P(\vec{m}|W)$ is given as a multivariate normal

$$P(\vec{m}|W) \quad \sim \quad N\left(\vec{\mu}_l = \frac{v\vec{\mu}_0 + l\bar{X}_l}{v+l}, (v+l)\,W\right) \tag{2.3}$$

where v encodes the strength of the prior in terms of an equivalent number of "prior" observations and l is the number of "new" observations in the data set D. The posterior of W is distributed itself as a $Wishart(\alpha + l, T_l)$, where α specifies the degrees of freedom the Wishart distribution (for this reason, $\alpha \geq n$ must be satisfied). $\bar{X}_l$ and S_l are the sample mean and covariance of D, and $\vec{\mu}_0$ and Σ_0 are the mean vector and covariance of the prior network structure. The matrices T_l and T_0 are calculated as

$$T_l = T_0 + S_l + \frac{vl}{v+l}\left(\vec{\mu}_0 - \bar{X}_l\right)\left(\vec{\mu}_0 - \bar{X}_l\right)' \quad (2.4)$$

$$T_0 = \Sigma_0 \frac{v\left(\alpha - n - 1\right)}{v+1}. \quad (2.5)$$

T_0 is the precision matrix of the of the prior marginal distribution on W (before the observed data D is introduced), given as $P\left(W\right) \sim Wishart\left(\alpha, T_0\right)$. The BGe metric scores the likelihood of an hypothesized network structure B_s given a data set D and the prior ξ as

$$P\left(D|B_s,\xi\right) = \prod_{i=1}^{n} \frac{P\left(D_{x_i,\Pi_i}|B_s^c,\xi\right)}{P\left(D_{\Pi_i}|B_s^c,\xi\right)}, \quad (2.6)$$

where $P\left(D_{x_i,\Pi_i}|B_s^c,\xi\right)$ is the local score of the data relevant only to variable, x_i, and its parents, Π_i. Specifically, we keep only those rows and columns of the T_0 and T_l that correspond to variables in $x_i \cup \Pi_i$ in the case of the numerator in (2.6). Similarly for the denominator, we keep only those rows and columns in T_0 and T_l corresponding to the variables in Π_i. The term B_s^c represents a fully connected Gaussian Network with edges among all variables. Both the numerator and denominator in (2.6) are calculated as follows:

$$P\left(D|B_s,\xi\right) = \left(2\pi\right)^{\frac{-nl}{2}} \left(\frac{v}{v+l}\right)^{\frac{n}{2}} \frac{c\left(n,\alpha\right)}{c\left(n,\alpha+l\right)} \left|T_0\right|^{\frac{\alpha}{2}} \left|T_l\right|^{-\frac{\alpha+l}{2}} \quad (2.7)$$

$$c\left(n,\alpha\right) = \left[2^{\frac{n\alpha}{2}} \pi^{\frac{n(n-1)}{4}} \prod_{i=1}^{n} \Gamma\left(\frac{\alpha+1-i}{2}\right)\right]^{-1} \quad (2.8)$$

To score an entire network, the expression in (2.6) must be evaluated or, equivalently, the expression in (2.7) must be evaluated for each variable in the domain and then each resultant value multiplied together. In the case of a nonuniform prior over network structures, an additional weighting of $P\left(B_s|\xi\right)$ should be factored into (2.6).

Provided with a scoring metric for Linear-Gaussian Bayesian Networks, we implement a simple hill-climbing algorithm to find a good structure for the networks. The algorithm is initialized with a prior network structure, then it takes one of the following actions:

(1) add an arc between variables x_i and x_j if no arc existed already; (2) remove an existing arc between two variables; or (3) reverse an existing arc between two variables. All three options are undertaken with the constraint that the resulting structure must remain acyclic. Each of the possible networks created by taking one of the these actions is evaluated using the BGe metric, and the action resulting in the highest scoring network is taken. The resultant network then becomes the initialization point for another application of this hill-climbing search. The process is repeated until taking a single action (adding, removing, or reversing an arc) creates no increase in the score, in which case an optimum has been reached. Unfortunately, this hill-climbing methodology is subject to local optima, and so the final network is perturbed. This perturbation is achieved by examining every existing edge in the current structure and, with some probability, removing the edge, reversing the direction of the edge (pending no cycle is created), or making no change. In cases where no edge exists between two variables, we consider adding an edge (pending no introduced cycle) or taking no action. The complete algorithm halts after performing a user-specified number of perturbations/restarts, and the the best-scoring network is returned. We outline the aforementioned hill-climbing algorithm in Algorithm 1.

It is important to note that structure returned from this hill-climbing search may not be unique relative to its score. The BGe metric demonstrates a property known as Score Equivalence, which means that it scores network structures belonging to the same Markov Equivalence Class (MEC) equally. An MEC is the set of graphs that represent the same set of conditional independence relationships between variables. Moreover, the algorithm described above only returns the structure of the network (i.e., the set of parent-child arcs); it does not compute the parameter values for each variable (means, variances, and regression weights). In Section 2.6.4, we describe how we arrive at these values by computing the Maximum Likelihood Estimates (MLE) for each parameter directly from the data set, D.

The BGe metric is considered a local scoring function because of its decomposition into summing over of node child/parent configurations. Specifically, if we consider taking the log likelihood of the probability in (2.6), we arrive at the following summation:

Algorithm 1 Hill-climbing with BGe Metric

Input: An initial Bayesian Network: B_{init}
Input: Number of perturbations to perform: $pturb$

$CurrentScore = BGe\left(B_{init}\right)$
$CurrentNet = B_{init}$
$BestScore = BGe(B_{init})$
$BestNet = B_{init}$
for $i = 1$ to $pturb$ **do**
  $LastScore = -\infty$
  while $CurrentScore > LastScore$ **do**
    $LastScore = CurrentScore$
    $AddNet = AddArc(CurrenNet)$
    $RemNet = RemoveArc(CurrentNet)$
    $RevNet = ReverseArc(CurrentNet)$
    $NextNet = \text{argmax}_{net}\left[BGe(AddNet), BGe(RemNet), BGe(RevNet)\right]$
    if $BGe(NextNet) > CurrentScore$ **then**
      $CurrentScore = BGe(NextNet)$
      $CurrentNet = NextNet$
    end if
  end while
  if $CurrentScore > BestScore$ **then**
    $BestScore = CurrentScore$
    $BestNet = CurrentNet$
  end if
  $CurrentNet = Perturb(CurrentNet)$
  $CurrentScore = BGe(CurrentNet)$
end for
return $BestNet$

$$\log P\left(D|B_s,\xi\right) \;=\; \sum_{i=1}^{n}\left[\log P\left(D_{x_i,\Pi_i}|B_s^c,\xi\right) - \log P\left(D_{\Pi_i}|B_s^c,\xi\right)\right]. \tag{2.9}$$

We can then compute the log likelihood equivalent of expression Equation 2.7 to compute the terms within the summation.

$$\begin{aligned}
\log P\left(D|B_s,\xi\right) &= \log\left[(2\pi)^{\frac{-nl}{2}}\left(\frac{v}{v+l}\right)^{\frac{n}{2}}\frac{c\left(n,\alpha\right)}{c\left(n,\alpha+l\right)}\left|T_0\right|^{\frac{\alpha}{2}}\left|T_l\right|^{-\frac{\alpha+l}{2}}\right] && (2.10)\\
&= \left[\frac{-nl}{2}\log\left(2\pi\right)\right] + \left[\frac{n}{2}\log\left(\frac{v}{v+l}\right)\right] && (2.11)\\
&+ \left[\log c\left(n,\alpha\right) - \log c\left(n,\alpha+l\right)\right] + \left[\frac{\alpha}{2}\log\left|T_0\right|\right] && (2.12)\\
&+ \left[-\frac{\alpha+l}{2}\log\left|T_l\right|\right] && (2.13)
\end{aligned}$$

Once in summation form, it becomes clear that computing the BGe score for an entire network is only necessary for the initial prior network, B_{init}. Each subsequent change in the network structure by adding, removing, or reversing an arc only requires a modification of some factor in the score in (2.9). For example, if we add an arc from variable x_i to x_j, then only the parent set of x_i has changed; consequently, we only need to recompute the i^{th} term in (2.9). The score for the resulting network would be the original network score minus the original i^{th} term (the one *not including* x_j as a parent) plus the new i^{th} term (the one *including* x_j as a parent).

Figure 2.5 contains a learned structure for the FishNet deployment. The initial prior network assumed complete independence among all six sensor stations at the site and placed a standard Normal distribution over all sensors (mean of 0 and variance of 1.0). The training set is constructed from observations from the SensorScope stations themselves. As we have no ground-truth data for the true temperature variables (X_i's), we consider those observations not excluded by the website range-checker or representative of a flatline sensor failure (i.e., consecutive -1 ℃ values) to be the "true" temperature values. The structure was learned using data from the first half of the deployment; however, the training set was limited to only those observations where all sensors reported a

value. If a sample missed at least 1 observation (one sensor failed to report within the 10-minute window), then it was rejected.

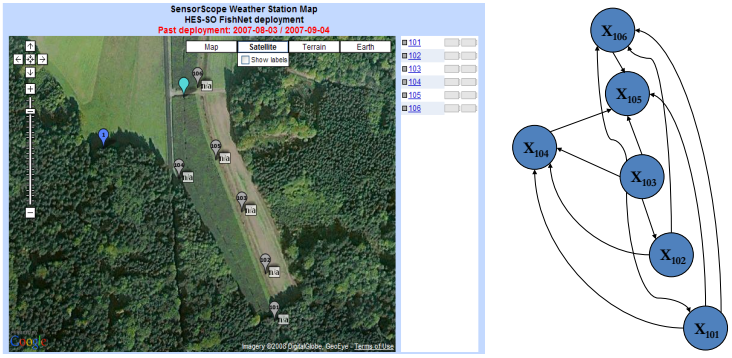

Figure 2.5: Left: Top-down view of the FishNet Deployment. Right: Learned dependency relationships among the six sensor stations at the deployment.

2.6.2 Incorporating a Temporal Model

The spatial model, learned from the methods described in the previous section, captures much of the correlative relationship among multiple sensors within a given deployment; however, the model suffers from some significant drawbacks. Foremost, the spatial model ignores the transition dynamics present in ecological data – a single sample taken from all sensors in a time step is considered independent of all temporally nearby samples (the sample taken 10 minutes ago, 20 minutes ago, etc.). Many types of ecological data are highly autocorrelated. In the case of air temperature, the diurnal cycle and seasonal cycle mean that data observed 24 hours and 365 days in the past, respectively, tend to correlate with data observed now. Because we generally do not know (or cannot observe due to limited deployment durations) the existence of all cycles in the data a priori, we implement only a first-order Markov relationship in the process model to insure that all stations transition from the current observation period (10 minutes, for example) to the next in a consistent manner. This is achieved through the introduction of a parent lag variable for each "true" temperature variable in the spatial model. The lag variables capture the state of the process in the last time step.

The addition of a Markovian lag changes our model from a static Bayesian network

into a dynamic Bayesian network. Conceptually, we can now imagine our learned spatial model being repeatedly "stamped-out" over the course of l samples in our database. Each stamp, or layer, contains the learned Bayesian network representing the spatial relationships in addition to a lag variable for each sensor. Figure 2.6 depicts the temporal model appended to the learned spatial model for the FishNet deployment.

As mentioned in Section 2.6.1, the spatial component returned by our hill-climbing search is not a unique representation of the set of conditional independencies between the temperature variables; it belongs to a Markov Equivalence Class. However, once we append our temporal model to each network in the MEC, the resultant models may no longer belong to the same class. To choose the best candidate network for the spatial model, we generate all members of this set by using an approach described by Andersson et al. [2]. Given a directed acyclic graph (DAG), their algorithm returns an *essential graph* that represents the equivalence class and has directed or undirected edges in place of all edges in the input graph. Undirected edges represent relationships between variables that can be reversed (parent becomes the child and visa versa) without changing the overall set of conditional independence relationships. We input our learned spatial model to create the essential graph representing its MEC. We consider all permutations of orientations for the undirected edges in the essential graph such that no cycles are introduced. For each DAG generated in this fashion, we append a set of lag variables, fit the parameters of the combined model as described in Section 2.6.4, and score the likelihood of the training data given this new network. We choose the spatial model that yields the highest data likelihood when combined with the temporal component.

The first-order Markovian assumption means that we need only consider the state of the process in the previous time step and observations in the current time slice when inferring the posterior distribution over the current state of the process. For example, to compute the posterior of X_{104} at time t from the DBN in Figure 2.6, we need only the distribution over the previous time step and any observations in the current time step

$$P\left(X_{104}^{t}|X_{101}^{1},X_{101}^{2},...,X_{101}^{t-1},X_{102}^{1},X_{102}^{2},...,X_{102}^{t-1},...,X_{106}^{1},X_{106}^{2},...,X_{106}^{t-1}\right) \quad (2.14)$$

$$= P\left(X_{104}^{t}|X_{101}^{t-1},X_{102}^{t-1},X_{103}^{t-1},X_{104}^{t-1},X_{105}^{t-1},X_{106}^{t-1}\right) \quad (2.15)$$

Each variable in our original network now has one additional parent variable and thus one additional parameter (weight associated with the new parent) to estimate. We

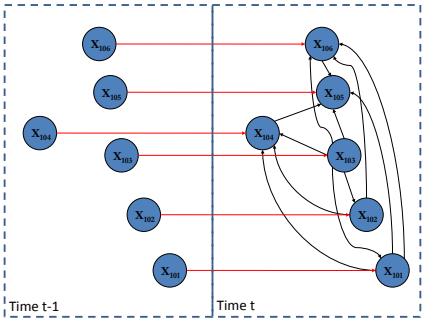

Figure 2.6: Time slices are designated by the dashed rectangles. Lag variables are appended for each sensor in the deployment, representing the state of the process in the last time slice.

can still apply our MLE technique for estimating the new parameter values; however, the training set for this model is now a subset of the training set used for our spatial model. This is because we now must place the additional constraint on our training data that it consists of contiguous pairs (two consecutive 10-minute intervals) where all sensor observations are present. We discuss how we respect this constraint in our Experiments and Methodology section.

2.6.3 Incorporating the Sensor Model

The combined spatial and temporal model represents the transition dynamics of the process over time, as well the correlative structure between the different sensor stations within a deployment. However, we cannot track the progression of the process without external observations; to this end, we incorporate a sensor model that represents the state of the sensor at each time slice and the observation recorded at that station. We represent the sensor state with a discrete variable, S_i, for SensorScope station i that can assume one of two values $S_i \in \{working, broken\}$. The sensor observation is represented as another Normally distributed variable conditioned on the state of the sensor and the current estimate of the air temperature as given by our process model. We will denote the observed variables as O_i where i refers to sensor i within the deployment. Figure 2.7 represents an abstract visualization of the combined spatial, temporal, and sensor models.

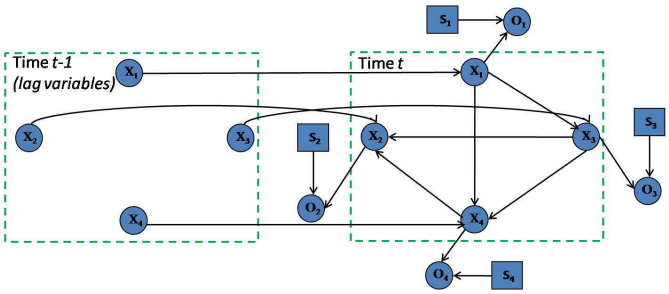

Figure 2.7: Time slices are designated by the dashed rectangles. The variables within the dashed area an abstract representation of the learned spatial model among four sensor variables. A sensor state variable and an observation variable are attached to each of the four sensor variables in the current time slice.

We manually set the parameters of the sensor state variables and observation variables. Again, this manual tuning is necessary because there are no known labels for the sensor states in any of the SensorScope datasets. For each S_i, we set the $P(S_i = working) = P(S_i = broken) = .5$ and for O_i

$$P\left(O_i|S_i = working, X_i = x_i\right) \sim N\left(x_i, 0.1\right) \text{ and} \tag{2.16}$$

$$P\left(O_i|S_i = broken, X_i = x_i\right) \sim N\left(.0001x_i, 10000.0\right). \tag{2.17}$$

This parameterization stems from the idea that the sensor state must be able to "explain away" the discrepancy between the observation variable, O_i, and the current estimate of the true air temperature, X_i. That is, if the sensor is believed to be working, then the observation value should be equal to that of the process model's estimate with some small, additional variance (0.1 ℃); contrarily, if the sensor is believed to be broken, then the observation has little do with the actual process and so is much noisier (10000.0 ℃ variance). The 0-mean, large variance distribution of the *broken* state approximates a uniform distribution over the possible range of observed sensor values. If we had ground-truth labels for the sensor state in each observation, explicitly modeling each fault with a separate distribution would not help us identify new anomaly types not seen in the training data. However, we could estimate $P(S_i = working)$ as the ratio of the number of working sensor observations over the total number of observations (and $P(S_i = broken) = 1 - P(S_i = working)$).

2.6.4 Parameter Estimation

Recall that under the assumption of a linear-Gaussian model, a Normally distributed variable, $X \sim N(\mu_x, \sigma_x^2)$, conditioned on a Normally distributed parent, $Y \sim N(\mu_y, \sigma_y^2)$, has the following density function (assuming both are univariate):

$$P(X|Y) = \frac{1}{\sigma_x\sqrt{2\pi}} \exp -\frac{(x-(w_1 y + \mu_x))^2}{2\sigma_x^2} \tag{2.18}$$

That is, $P(X|Y) \sim N(\mu_x + w_1 y, \sigma_x^2)$, where w_1 is a scalar weight multiplied with an input, y, drawn from Y's distribution.

Once our structure learning algorithm has provided each variable in our domain with a set of parents (including the temporal lag variables), the MLE approach to estimating the values of the parameters (μ_i, σ_i^2, and w_i) reduces to solving a multiple linear regression problem [61]. Specifically, we solve

$$\hat{\theta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\vec{Y}, \tag{2.19}$$

where $\hat{\theta}$ represents the mean and associated weights of the target variable (the variable whose parameters we are currently estimating), $\mathbf{X}$ is a matrix containing the value of the parents of the target variable in the data set across all samples, and $\vec{Y}$ is a vector containing all the values of the target variable corresponding to the inputs in $\mathbf{X}$.

2.6.5 Inference

Inference is performed in our models using the Variable Elimination (VE, [15]) algorithm adapted for Conditional Linear Gaussian models [39, 40, 51]. There are two inference queries made at each time step, t.

First, we wish to compute the maximum a posteriori (MAP) assignment of the discrete sensor variables, $\vec{S}^t$, given the set of sensor observations, $\vec{O}^t$,

$$P\left(S_1^t, S_2^t, ..., S_n^t | O_1^t = o_1^t, O_2^t = o_2^t, ..., O_n^t = o_n^t\right). \tag{2.20}$$

This requires marginalization of the the hidden "true" temperatures (continuous variables)

1. Begin with initial Bayesian Network structure, B_{init}, for the input data, D. If no initial network is provided, B_{init} is a network containing no arcs and each variable $x_i \in B_{init} \sim N(0, 1.0)$
2. Compute the sample mean and covariance of D, $\bar{X}_l$ and S_l.
3. Compute the mean and covariance represented by B_{init}: $\vec{\mu}_0$ and Σ_0.
4. Compute T_0 and T_M using the values from steps (2) and (3) in equations (2.5) and (2.4).
5. Perform hill-climbing (Algorithm 1) initialized from B_{init}. Call the resultant structure B_{post}.
6. Build the Markov Equivalence Set, $\{B_k | B_k \in \text{MEC}(B_{post})\}$ and append the temporal model to each network B_k.
7. Compute MLE parameters for each $B_k \in \text{MEC}(B_{post})$ from the data, D.
8. Compute $B_{best} = \arg\max_{B_k} P(D|B_k)$.
9. Append Sensor State (S_i) and Observation variables (O_i) for each sensor variable (X_i) in B_{best}. The parameters of these variables are manually set.

Table 2.1: The table lists the order of steps in our structure learning algorithm for constructing a spatiotemporal process model.

at time t and $t-1$. The remaining sensor-state variables (discrete) are contained in a single potential whose distribution is represented by a table having an exponential number of entries. Each entry corresponds to one of the 2^n possible configurations of n sensor-state variables; consequently, construction of this table occurs in time exponential with the sensor count. The sensor counts in the deployments discussed herein were not prohibitively large; however, for deployments containing more sensors, we could consider approximate inference algorithms, such as Gibbs Sampling [25] or other particle filter methods. These algorithms approach the exact solution as the number of samples or particles used increases. While each sample can be generated in linear time, the number of samples required to reasonably approximate the true joint posterior may be exponential in the number of sensors. Alternatively, we could impose a prior on our spatial structures that would encourage learning disjoint spatial models (i.e. spatial models where one or

more of the X_i variables is disconnected from the remainder). In this case, exact inference would be exponential in the number of sensors in the largest subgraph.

Second, we treat the MAP assignment as new evidence for the sensor states at time t and compute the updated estimate of the hidden "true" temperatures, $\vec{X}^t$,

$$P\left(X_1^t, ..., X_n^t | S_1^t = s_1^t, ..., S_n^t = s_n^t, O_1^t = o_1^t, ..., O_n^t = o_n^t\right). \tag{2.21}$$

Because we now observe the sensor states, computing the posterior over the true temperatures becomes a query over a linear-Gaussian model. Variable Elimination takes cubic time in the number of sensors for this query and so is tractable to perform exactly. The posterior distribution on the true temperatures is passed forward as a message to be used in inference at time $t + 1$. The joint posterior distribution over the true temperature variables can be thought of as an α message in the forward pass of a filtering algorithm [59]. If the MAP estimate of the sensors at time t indicates that sensor i is working ($S_i = working$), then we input its corresponding observation (O_i) for the true temperature's lag variable at time $t+1$; otherwise, we use the corresponding α message to specify a distribution over the lag's value. We then repeat this two-step query procedure for time $t + 1$.

Our motivation for handling inference in this two-step process is that, in an online setting, we must make a decision that each sensor at time t is working or broken rather than postponing this decision and maintaining a "belief state," that is, 79% working and 21% broken. Not only is this approximation useful for an online QC system, it also exempts us from having to maintain an exact belief state that increases in size after each time step. To clarify, the exact belief state at time t would be a 2^{nt} component mixture of n-dimensional multivariate Gaussians. Once we have determined the state of each sensor, we need to propagate forward an α message regarding the true temperatures $\vec{X}$ to time $t+1$ (computed in (2.21)). Thus, our approximation is made by considering only the mixture component corresponding to the MAP of the S_i variables at each time step.

2.7 Experiments and Methodology

Our experiments focus on the validation of our learned spatial models across varying deployments and the efficacy of our complete DBN model as a tool for quality control.

We address each issue in turn. We perform the former validation through a series of hold-one-out prediction tests to determine the relative strength of multiple stations as a predictor for an individual missing station. Second, we provide a comparative analysis of the performance of three different QC models as applied to real data from the SensorScope project. The three models are the spatial, temporal, and spatiotemporal models already discussed, each augmented with a sensor model as described in Section 2.6.3. The experiments reflect the weaknesses and strengths of each model, and show preliminary justification for pursuing a spatiotemporal approach. Lastly, we evaluate the performance of our model in terms of type I and type II error rates. This experiment is performed via the addition of artificial noise to the original datasets in order to create labels that can be matched against our predictions of the sensor variable.

All experiments in this section were performed with a data set spanning from the beginning of the respective deployment to its end. Because the SensorScope stations are not necessarily synchronized to sample at the same time, the data was binned and averaged into 10-minute windows consistent across all stations. A training set and testing set were created for each deployment by roughly splitting the data into halves, in which the first half (representing the first chronological half) became the training data and the second half became the test set. In all experiments, only training samples (10 minute windows) where readings for all of the stations were present were used, and so often the training sets are significantly smaller than the testing sets. For experiments containing a temporal model, only those training samples that had a fully observed preceding sample (the last 10-minute period) were used. Data for the experiments comes from the FishNet and the Grand St. Bernard deployments. Grand St. Bernard was a third deployment located in the Grand St. Bernard Pass between Switzerland and Italy (at an elevation of 2300 meters) and was in place from September 13, 2007 to October 26, 2007. All six sensors were used in the FishNet deployment; however, only a subset comprising 9 of the 23 stations were used from the Grand St. Bernard (see Figure 2.8). Because we are only including training samples where all sensor measurements are present, including all stations from the Grand St. Bernard would exclude too many potential samples.

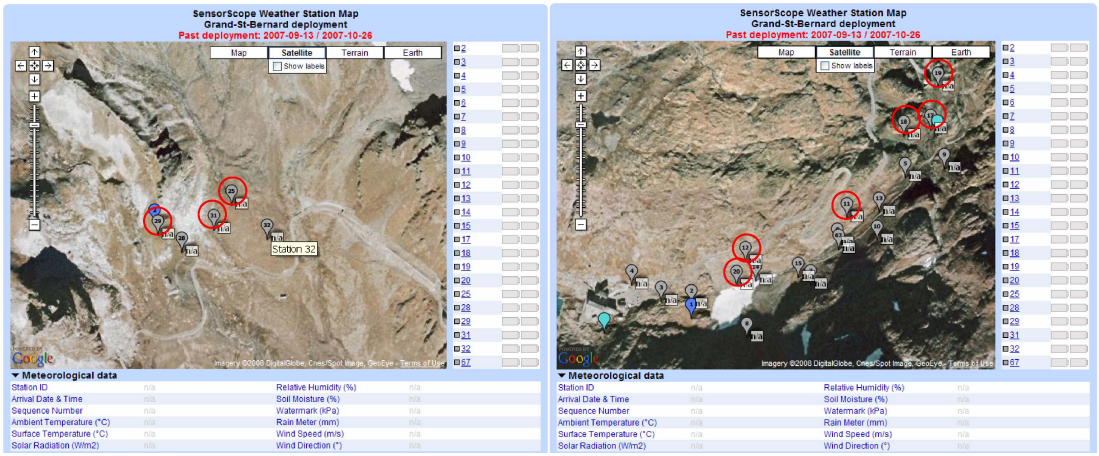

Figure 2.8: Left: The portion of the Grand St. Bernard deployment on the Italian side of the mountain pass. Right: The Swiss side of the Grand St. Bernard deployment, located approximately 2 kilometers east of the Italian deployment. The stations circled in red denote those stations chosen for purposes of modeling.

2.7.1 Leave-One-Out Prediction

The leave-one-out experiments are performed by withholding a sensor's observation and computing the posterior distribution over the hidden sensor value given all other sensor observations and the learned spatial (Section 2.6.1) and spatiotemporal models (Section 2.6.2). We report results for the FishNet and Grand St. Bernard SensorScope deployments. In both cases, the spatial model is learned and parameterized using only the first half of the data (approximately 1400 and 440 training samples, respectively).

Once the spatial and spatiotemporal models are trained, we process the testing data (second half of the collected samples) in an iterative manner. In each iteration, a single observation, representing the measurement taken at one station at one time point, is removed. We compute a posterior prediction for the removed value using the learned spatial model and the observations from all other stations in the case of the spatial model, and all other stations in addition to all measurements from the previous time step in the case of the spatiotemporal model. We compute the mean squared error (MSE) between the predicted value for the withheld observation and its actual value in the test set, as well as the variance in our prediction. Let $t = 1, ..., T$ denote the time (sample) index, $i = 1, ..., n$ index the "true" temperature variable X_i, and x_i^t be the value of the

true temperature at station X_i at time t. The MSE and Variance for station X_i is then computed as

$$MSE_i = \frac{1}{T}\sum_{t=1}^{T}\left(E\left[P(X_i|\mathbf{X}\backslash X_i)\right] - x_i^t\right)^2. \quad (2.22)$$

The variance of the posterior estimate of X_i depends only on the set of variables that are observed (included in the set $\mathbf{X}\backslash X_i$); not on the exact value of those observations. Thus, we need only examine $Var\left[P(X_i|\mathbf{X}\backslash X_i)\right]$ for any one of the t samples above to determine the variance. The leave-one-out error is also measured using cumulative log likelihood (CLL),

$$CLL_i = \sum_{t=1}^{T}\log P(X_i = x_i^t|\mathbf{X}\backslash X_i), \quad (2.23)$$

and is shown as the dashed horizontal line in Figures 2.9, 2.10, and 2.11. We then perform a further computation, removing an additional variable's observation from the testing data. We compute the cumulative error over the training data (sum log likelihood) in predicting our original target variable with one additional sensor's observation missing. Using the same notation as above, we compute this as

$$CLL_{i,j} = \sum_{t=1}^{T}\log P(X_i = x_i^t|\mathbf{X}\backslash\{X_i, X_j\}). \quad (2.24)$$

Each bar in Figures 2.9, 2.10, and 2.11 corresponds to the new CLL value after the variable X_j has been hidden. The purpose of removing a second variable X_j is to measure the contribution of second variable in predicting the value of the first removed variable X_i.

Figure 2.9 (upper left plot) indicates that station 101 was not only the most difficult to predict (MSE of .56 ℃), but also gained the least from the presence of other sensors. Additionally, removing the observations of station 104 resulted in the largest increase in error for station 101; however, even this effect was not particularly significant in comparison to removing any of the other remaining stations. The likely reason for this lack of correlation is due to station 101's position on the south edge of the deployment (Figure 2.5), near the wooded border. Station 104, its most similar station, is also located

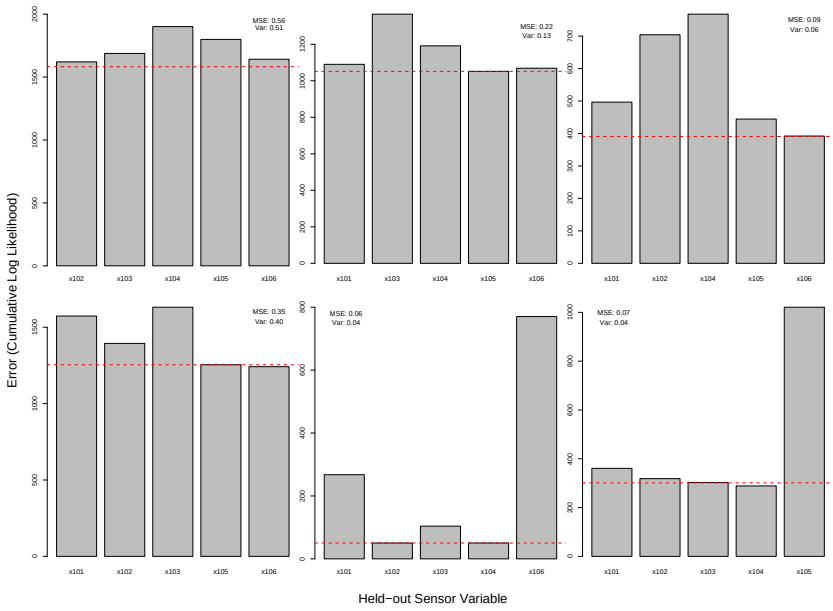

Figure 2.9: Redundancy Test for FishNet Spatial Model. Dashed line indicates the error in predicting the individual missing sensor. Each bar along the X axis represents the change in error from removing the additional sensor variable corresponding to that bar. The Y axis is the error measured as the cumulative log likelihood over all test cases of the true value given the predicted distribution.

in close proximity to a wooded, shady region, which may explain its role as the strongest predictor for station 101. This example highlights the fact that our "spatial" learning is discovering more correlation than those just based on spatial proximity as we might see in a Kriging model [44]. Rather, our model is capturing all sources of linear correlation between sensors at a given time step, without the use of a feature set describing each sensor.

Stations 105 and 106 (bottom center and bottom right plots, respectively) appear to be very highly correlated, as indicated by the dramatic increase in prediction error when either station is held out while predicting the other. Moreover, we see that when holding out each station (105 and 106), there is little error in reproducing the withheld observation given the presence of the other 5 sensors (MSEs of .058 ℃ and .074 ℃, respectively). The Sensirion SHT75 documentation reports a measuring accuracy of ±.35 ℃ in operating

conditions of 15.21 ℃ (average temperature of the FishNet site for the testing period).

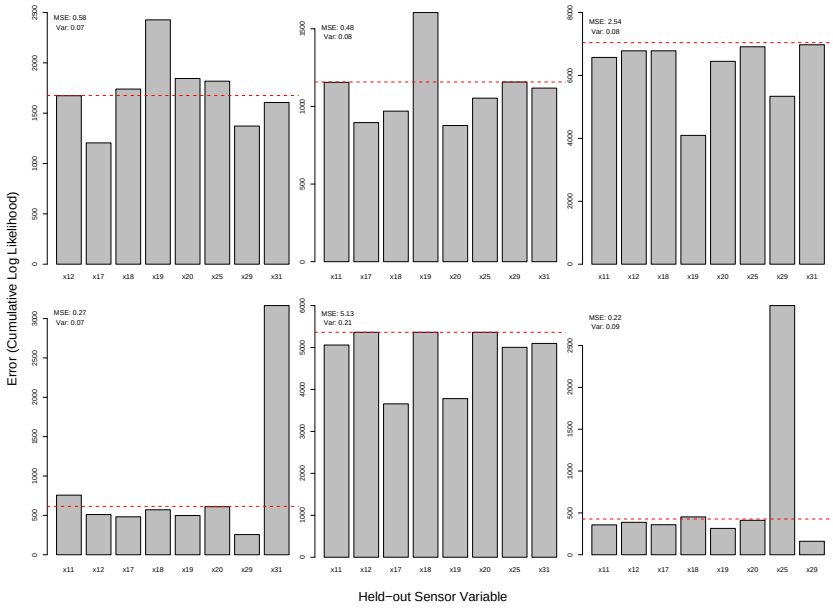

Figure 2.10: Redundancy Test for Grand St. Bernard Spatial Model. Dashed line indicates the error in predicting the individual missing sensor. Each bar along the X-axis represents the change in error from removing the additional sensor variable corresponding to that bar. The Y-axis is the error measured as the cumulative log likelihood over all test cases of the true value given the predicted distribution.

Figure 2.10 conveys a similar analysis performed with nine stations selected from the Grand St. Bernard deployment (Figure 2.8). The top row of bar plots depicts the analysis of stations 11, 12, and 17, while the bottom row corresponds to stations 25, 29, and 31. It is apparent from the plot that stations 17 and 29 are the most difficult to predict from the remaining 8 sensors. This stands to reason for station 29, for though it is located on the Italian side of the deployment with 25 and 31, it is still separated by a steep hillside dividing the region. We could not ultimately discern the reason for station 17's discordant behavior from the remaining sites. The Sensirion SHT75 documentation reports a measuring accuracy of ±1.0 ℃ in operating conditions of 1.83 ℃ (average temperature of the Grand St. Bernard site for the testing period).

It is interesting to note in Figure 2.10 that, in all cases, there exists at least one

sensor whose removal actually seems to decrease the amount of error in predicting the hold-out sensor's value. This trend suggests that our learned model may have overfit the original training data, and thus poorly generalized to the test set. In the case of air temperature data measured over 1-2 months (especially during seasonal transitions), data monitored at the beginning of the observation period can differ significantly from data measured toward the end of the observation period. This compounds the difficulty of our work, as now our underlying assumption of a single generative multivariate distribution creating our training and testing data is no longer valid. Future work will need to focus on time-series analysis techniques that can map the test set to our training set without full knowledge of the trend effects that shape the generative distribution over time.

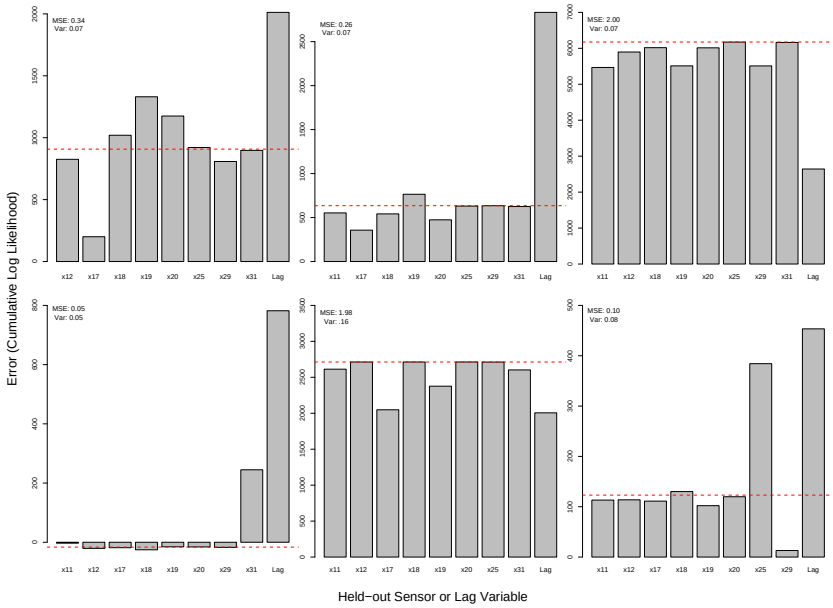

Figure 2.11: Redundancy Test for Grand St. Bernard Spatiotemporal Model. Dashed line indicates the error in predicting the individual missing sensor. Each bar along the X-axis represents the change in error from removing the additional sensor or lag variable corresponding to that bar. The Y-axis is the error measured as the cumulative log likelihood over all test cases of the true value given the predicted distribution.

Finally, Figure 2.11 shows the hold-one-out analysis applied to a spatiotemporal model learned from the Grand St. Bernard data. Recall that this model is simply the original

spatial model with the relevant lag variables appended to its structure, and the parameters reestimated to account for the additional set of parents. We notice that, in all cases, the hold-one-out error decreases with the incorporation of a lag effect. Also significant is that, with the exception of stations 17 and 29, the lag effect has the greatest predictive power for every station. This makes intuitive sense, as air temperature is unlikely to change significantly over the course of 10 minutes (the duration of the lag). Stations 17 and 29 suffer from the same overfitting problem in this revised model, as hiding the Markovian variable reduces error in both cases. In fact, the large magnitude of the gain incurred from hiding the lag variable from station 17 seems to support the theory that the nature of the correlative effect changed drastically between the training and testing periods. If it had not, then it is unlikely that the spatiotemporal model would have lent the lag variable such significant weight based on the training data.

In addition to providing some intuition about the values of parameters and network structures learned in the spatial component of our QC system, this type of hold-one-out analysis can be used to identify redundant sensors. For purposes of quality control, two sensors measuring the same phenomenon (or one able to near-perfectly predict the other's missing value) is necessary to truly validate recorded observations; however, for purposes of capturing all the heterogeneity encompassed within a site, it may be preferable to relocate any sensor considered redundant. This analysis can be easily generalized to hold-two-out in order to detect clusters of 3 sensors where one sensor can accurately predict the value of the other missing two (redundancy at this level would even be unnecessary for QC purposes).

2.7.2 Quality Control Experiments

We begin this section by providing a comparative analysis of the spatial-only and temporal-only models. Every model type discussed here contains the sensor state model, as described in Section 2.6.3, appended to the structure. That is, the spatial-only model is a learned network structure over the first half of the data collected from the deployment with a discrete sensor-state variable and a continuous, Normally distributed observation variable added for each sensor. The temporal-only model assumes independence between all stations, but contains an additional lag variable for each sensor and is auto correlated with that lag. Figure 2.12 demonstrates the performance of the spatial model as applied

to the second half (testing set) of the Grand St. Bernard deployment data.

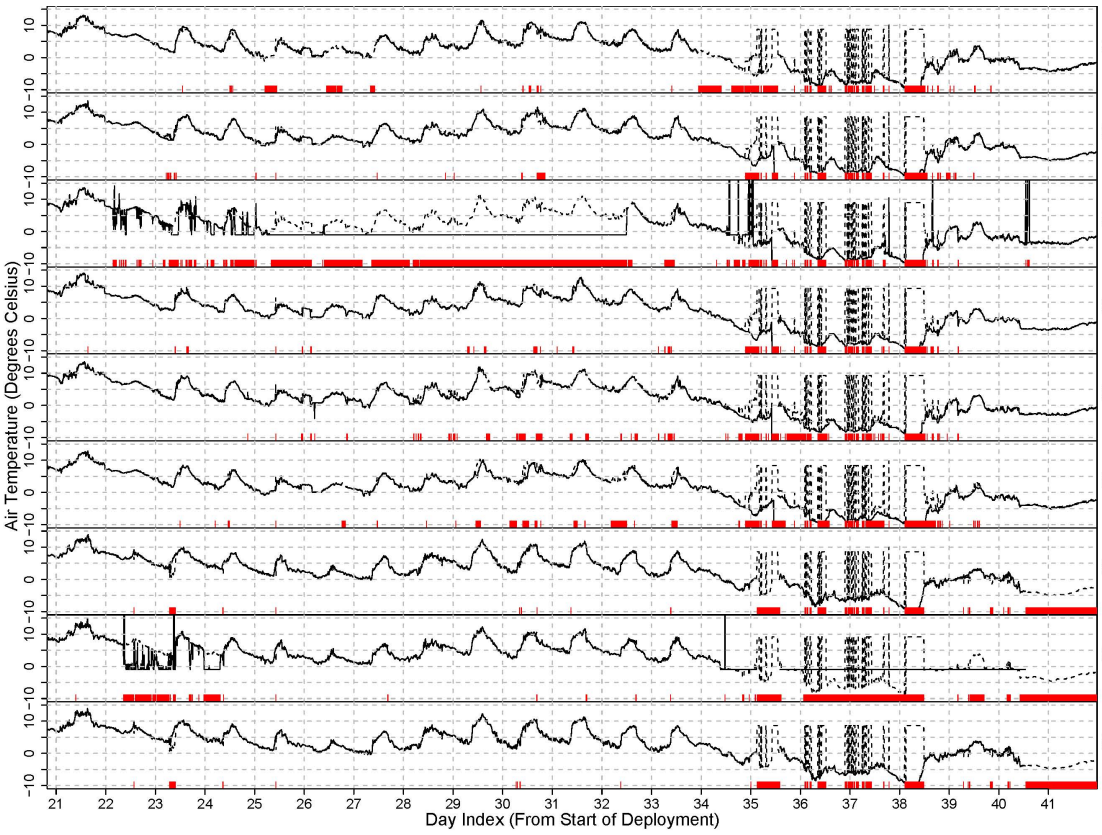

Figure 2.12: Quality Control performance on Grand St. Bernard using the spatial-only model. Solid line indicates the actual temperature recorded at each station. Dashed line indicates the posterior prediction made for that station. Red hashes at the base of each graph indicate a value labeled as anomalous (i.e., $S_i = broken$) by our system for that time period. The X-axis denotes the day (vertical dashed line depicts midnight) since the deployment began, and the Y-axis denotes temperature in degrees. Corresponding station names appear on the right side of the graph next to the stream that they depict.

The spatial-only model is able to recognize the spiky, anomalous behavior observed in both stations 17 and 29 between days 21 and 25. Moreover, this QC system detects the flatline anomaly when station 17's air temperature sensor reported 0-voltage, which is by default converted to a reading of -1 ℃. The dashed line represents the system's prediction of the actual temperature value and appears to be consistent with the neighboring stations

of sensor 17. Unfortunately, the lack of a temporal connection means that this model's behavior is static over time. The overall mean of each station remains constant, because there is no transition function to allow the mean of the process model to track the true temperature, nor is there any explicit conditioning on the seasonality or index in the diurnal cycle. The end result is that, as the mean of the true temperature begins to decrease to the point where it significantly differs from the learned mean in the training data, the model labels these new values as anomalous. This begins to manifest itself at day 35. Each time the average reading from all 9 sensors drops significantly below the training data mean, an anomaly is raised and the model imputes the training data mean as the correct value. The large disparity between the model's prediction and the actual observations between days 35 and 39 results in most of the observations therein being misclassified as anomalous, save for daytime high values.

Figure 2.13 shows the performance of the temporal-only model on the same Grand St. Bernard data. This model is equivalent to n disjoint Kalman Filter models [35], with an additional discrete sensor-state variable that explains away any discrepancy between the observation and predicted value of the air temperature. Of immediate note is that the -1 ℃ flatline in station 17 is no longer properly flagged as anomalous. A few nominal observations near -1 ℃ beginning on day 25 confuse the temporal model into tracking this flatline behavior. If the transition between temperature observations over time is gradual enough, the temporal-only model will track the temperature signal through periods of anomalous readings caused by sensor malfunction. Without external observations from correlated stations, the independent sensor cannot differentiate between slow changes in the observations due to a change in the process signal (warming or cooling trends) or the breakdown of the sensor. In cases where the observed value disagrees with the model's predicted value (the model loses tracking), future predictions drift toward the training data mean. This can be seen at station 11 on day 34 when the signal is completely lost, or station 17 on day 35 when an erratic spike followed by a drop in temperature throws off the model.

The temporal model's ability to track the process even as it drifts away (albeit slowly) from the trained mean gives it an advantage over the spatial-only model. We can see this in stations 12, 18, 20, 25, and 31, where a slow cooling effect does not disrupt the model's ability to track the process during the second half of the training period. Unfortunately, the assumption of complete independence between stations means that the model cannot

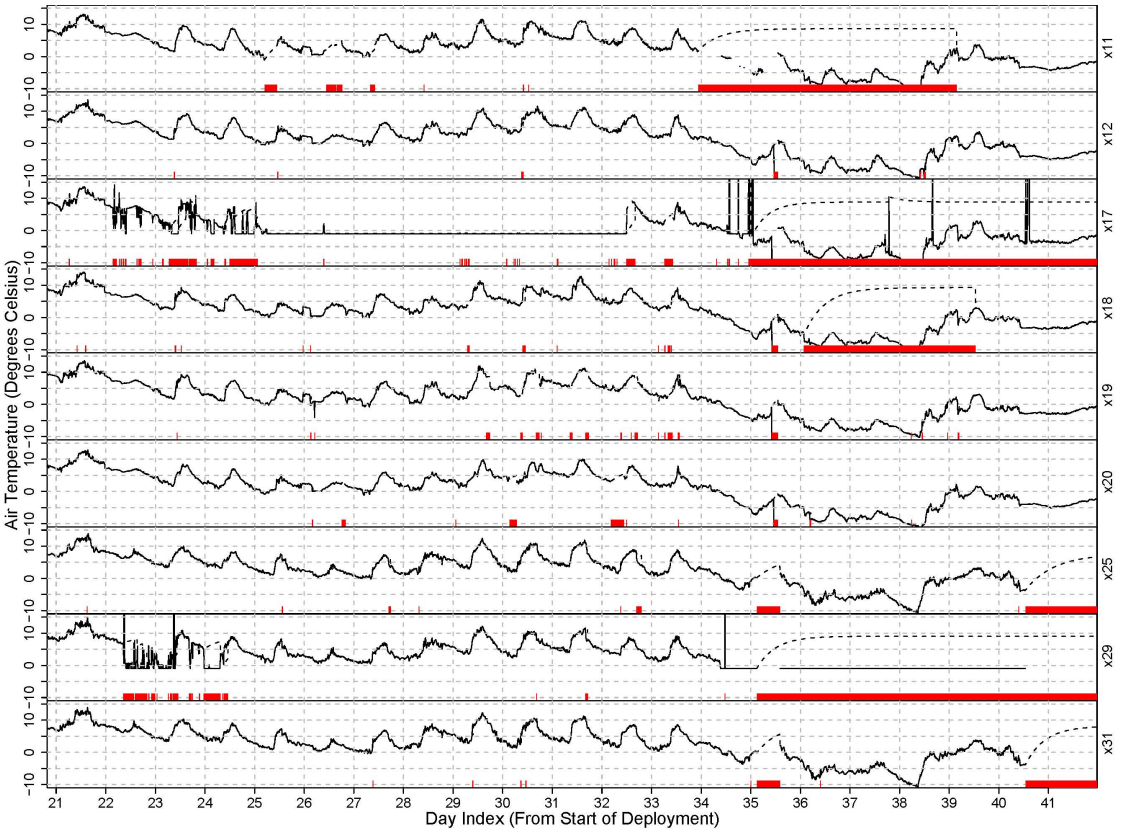

Figure 2.13: Quality Control performance on Grand St. Bernard using the temporal-only model. Solid line indicates the actual temperature recorded at each station. Dashed line indicates the posterior prediction made for that station. Red hashes at the base of each graph indicate a value labeled as anomalous (i.e., $S_i = broken$) by our system for that time period. The X-axis denotes the day (vertical dashed line depicts midnight) since the deployment began, and the Y-axis denotes temperature in degrees. Corresponding station names appear on the right side of the graph next to the stream they depict.

accurately reconstruct the true value of the temperature at stations diagnosed as *broken*, as seen in stations 11, 25, 29, and 31. To this end, we turn to the spatiotemporal model.

The performance of the spatiotemporal model (Figure 2.14) appears robust to the weaknesses in the spatial-only and temporal-only models. In particular, it is able to both detect and reconstruct the anomalous values from flatlined senors (station 17, days 25 to 33) and missing values (station 29, days 35 to 42). Further, the model permits some

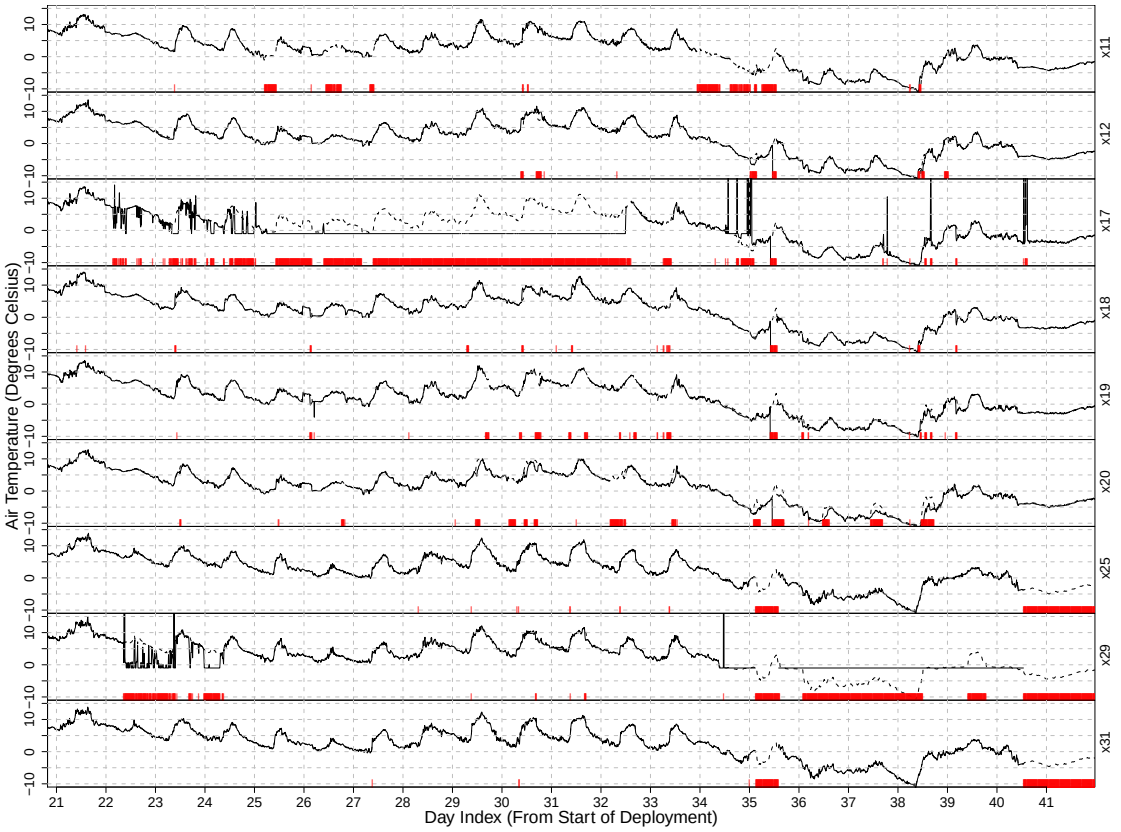

Figure 2.14: Quality Control performance on Grand St. Bernard using the spatiotemporal model. Solid line indicates the actual temperature recorded at each station. Dashed line indicates the posterior prediction made for that station. Red hashes at the base of each graph indicate a value labeled as anomalous (i.e., $S_i = broken$) by our system for that time period. The X-axis denotes the day (vertical dashed line depicts midnight) since the deployment began, and the Y-axis denotes temperature in degrees. Corresponding station names appear on the right side of the graph next to the stream they depict.

drift in the original learned distribution of the process model, as indicated by its accurate tracking of the air temperature from days 35 through 39, with few apparent false positives. Like the spatial and temporal models, the combined model is able to diagnose the obvious spikes in air temperature that are also indicative of sensor malfunctions (station 17, days 35 to 41). The overall false positive rate seems minimal (save for the midday periods on days 36-38 at station 20, where the predicted estimates are slightly higher); however,

without ground-truth data, we cannot determine the true type I and II error rates.

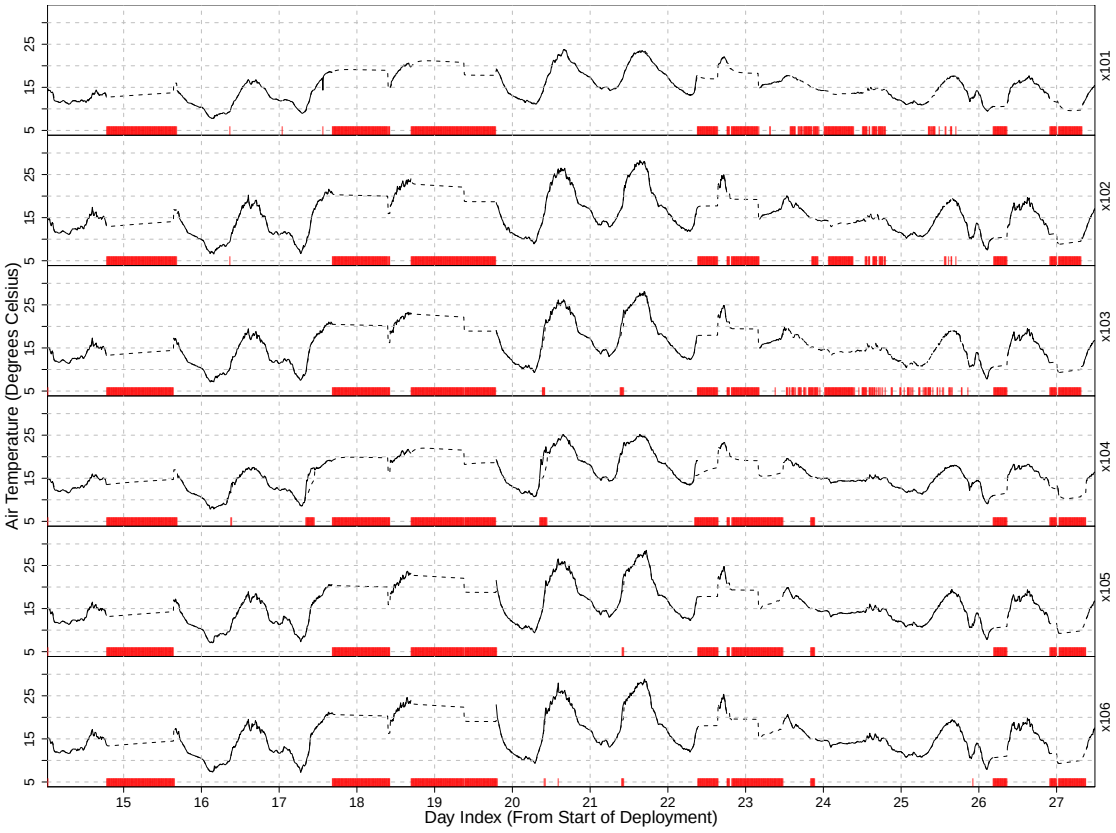

Figure 2.15: Quality Control performance on FishNet using the spatiotemporal model. Solid line indicates the actual temperature recorded at each station. Dashed line indicates the posterior prediction made for that station. Red hashes at the base of each graph indicate a value labeled as anomalous (i.e., $S_i = \textit{broken}$ by our system for that time period. The X-axis denotes the day (vertical dashed line depicts midnight) since the deployment began, and the Y-axis denotes temperature in degrees. Corresponding station names appear on the right side of the graph next to the stream that they depict.

Unfortunately, in cases where GPRS is lost (either due to a required reboot, or a failure at the station) and all station signals are lost, our spatiotemporal model cannot recreate the missing values. This is apparent in the FishNet dataset, where GPRS outages were relatively frequent (Figure 2.15) compared to Grand St. Bernard. In cases where no sensors are providing observations, the variance over the joint posterior of the current

process state grows. This is because a lack of evidence about the state of the process means that we are becoming increasingly uncertain about its current value. Eventually, the variance will grow so large that almost any observed value will seem likely, and so our process will begin to retrack the observations. When the sensor readings resume, the very first observation is recognized as nonanomalous and causes the spatial and temporal components to shift the process model to that observed value for all correlated sensors. For example, during the outage beginning on day 18.5 and lasting until day 19.75, there is a single observation around day 19.5 that causes the collective temperature prediction for most stations to shift down 4 ℃. Otherwise, during these periods of prolonged GPRS outage, the prediction will drift toward the mean for each station while variance on the prediction grows larger.

Without knowledge of neighboring observations within a site or a sufficiently high-order Markovian model, there is little the current model can do to correctly track the air temperature during periods in which all sensors fail to report. One potential solution would be to introduce a baseline calculation that represents prior knowledge about the air temperature at a given site for each time of day and time of year. In the absence of evidence to correct this baseline estimate, the model would default to the baseline values. While the baseline may be inaccurate given the temporal context (warmer/cooler than usual, storm effect, etc.), it would likely guide the process model such that it was closer to the actual signal when observations recommenced. Formulation of this baseline and its performance in a long-term stationary QC domain is reported in Dereszynski and Dietterich [17]. The problem remains that, in the short-term ecological monitoring setting, it may be difficult to estimate this baseline in the absence of a full cycle of the observed phenomenon (one year in the case of air temperature).

2.7.3 Noise Injection Experiments

To obtain a quantitative assessment of the accuracy of our quality control methods, we performed a series of experiments in which we injected noise into the SensorScope data. Initially, all readings are assigned to be nonanomalous. Then any missing values or 10-minute average of exactly -1.0 degrees ℃ are labeled as anomalous. Finally, synthetic faults are introduced by taking each data point and, with probability η, adding a noise value drawn from a normal distribution with zero mean and variance σ_n^2. Each synthetic

fault is labeled as anomalous.

We report the results of the noise injection experiments for both the FishNet and Grand St. Bernard deployments. The experiments were performed using values of σ_n^2 ranging from 3 ℃ to 30 ℃ in increments of 3 ℃ and values of η ranging from 5 to 50 percent in increments of 5 percent. Thus, we evaluate 100 different variations on the testing data for each dataset, and record the results in terms of the number of true positives (TP, number of anomalous data values classified by our system as such), true negatives (TN, the amount of clean values that were not flagged by our system as anomalous), false positives (FP, misclassified anomalies that were actually clean), and false negatives (FN, values that were actually anomalous, but not detected by our system). The results are summarized in terms of Cohen's κ statistic (the rate of agreement between our classifier and the true labels correcting for chance agreement [8]), precision (the total number of true positives divided by the number of true positives plus false positives), and recall (the total number of true positives divided by the number of true positives plus false negatives). Cohen's κ reflects the degree to which our algorithm reproduces the true labels as created by our noise injection process corrected for chance predictions [8, 68]. It is calculated as

$$\kappa = \frac{P(O) - P(E)}{1.0 - P(E)}, \tag{2.25}$$

$$P(O) = \frac{TP + TN}{FN}, \tag{2.26}$$

$$P(E) = \frac{TN + FP}{N} \times \frac{TN + FN}{N} + \frac{FP + TP}{N} \times \frac{FN + TP}{N}, \tag{2.27}$$

where $P(O)$ is the observed probability of the classifier agreeing with the true label, $P(E)$ is the expected probability of chance or coincidental agreement, and N is the total number of samples in our testing data. Regarding the latter two statistics, precision provides a sense of how many of the values we label as anomalous are truly indicative of sensor faults, while recall summarizes what percent of the total genuine sensor faults we detect in the data. In this application, we are interested in achieving as much precision as possible at high levels of recall. In other words, we want to make sure we detect most of the sensor faults, even if this leads to some false alarms (false positives).

Let us consider what results we should expect from injecting noise. First, we would

expect that as the magnitude of the noise (σ_n^2) increases, the noise will become easier to detect, because it will be clearly distinct from nearby values in time and space. Increasing the amount (η) of noise should not decrease our ability to detect it; however, fewer non-noisy values will make tracking short-term changes in the air temperature (due to storm effects, cold/warm fronts, etc.) more difficult. For this reason, we expect that larger values η will result in more false positives in cases where the model loses tracking and its predictions drift away from the true air temperature. Ultimately, the best data-anomaly detection performance will be obtained when there is a small amount of very obvious noise in the data (small η and large σ_n^2).

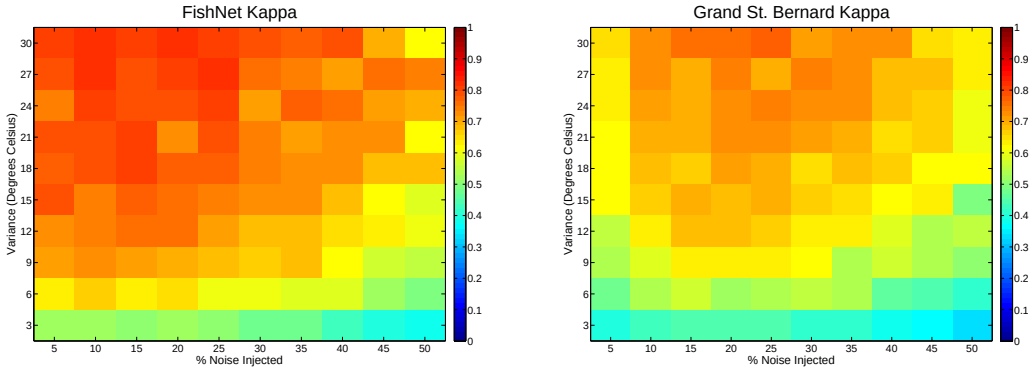

Figure 2.16: κ as a function of both noise variance (Y-axis) and percentage of data points modified by noise (X-axis). The shade of color associated with each grid cell reflects the degree of κ (on a scale of 0 to 1.0) for that configuration of noise level and saturation.

Figure 2.16 shows the κ rates of our model applied to the FishNet (left) and Grand St. Bernard (right) deployments. The value of κ is displayed on a color scale, with higher values shown in darker shades of red and lower values shown in darker shades of blue. As expected, larger values of σ_n^2 resulted in better κ scores for both the FishNet and Grand St. Bernard data sets. Data anomalies drawn from a higher variance distribution are more evident to our classifier; consequently, there is more genuine agreement. In the of case FishNet, κ increases from .527 to .826 as we increase σ_n^2 from 3 to 30, and from .440 to .755 in the case of Grand St. Bernard (at $\eta = 20\%$). Interesting to note is that more noise in the data does not have an adverse effect on our κ scores until more than 25% to 30% noise is introduced. In fact, the κ scores for both data sets increase up to

this point. As more anomalies are introduced into the data and correctly identified by our algorithm, the likelihood of coincidental agreement ($P(E)$) decreases; however, there appears to be a threshold at approximately 25% noise where κ begins to decrease. This suggests a tradeoff where further abundance of anomalies in the data makes them appear haphazard rather than systematic.

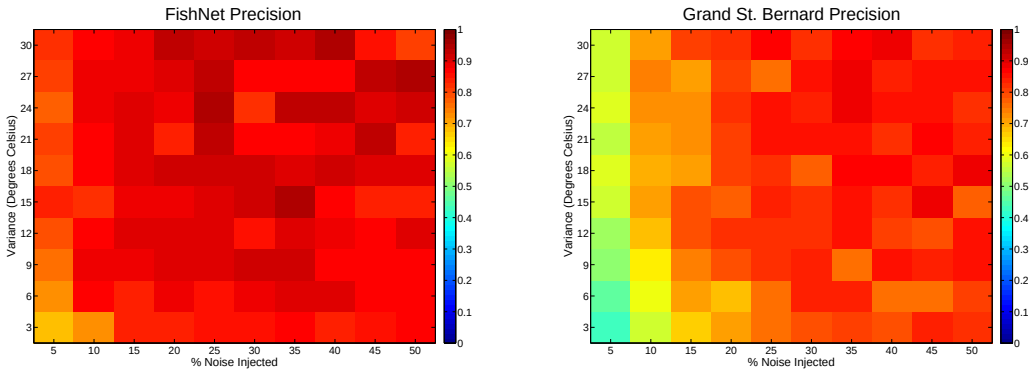

Figure 2.17: Precision as a function of both noise variance (Y-axis) and percentage of data points modified by noise (X-axis). The shade of color associated with each grid cell reflects the degree of precision (on a scale of 0% to 100%) for that configuration of noise level and saturation.

Precision results in Figure 2.17 further support our hypothesis regarding the effect of larger values for σ_n^2. The true anomalies are more disparate from the normal data, and so the ratio of true positives to false positives increases directly with magnitude of the noise (fewer false positives result in higher precision scores). Further, as we increase the amount of noise in the data, any value we classify as anomalous has a higher chance of actually being so due to a greater proportion of the data containing noise. When there are few anomalies in the data (5% injected noise), our scores suffer due to the presence of relatively many false positives. Consider our false positive rates (percent of all "good" data missclassified as anomalous) shown in Figure 2.18. Though we achieve very low false positive rates at this level of noise (average of 1.01% at FishNet and 3.08% at Grand St. Bernard for $\eta = 5\%$), there are too few synthetic anomalies in the data, causing the false positive counts to dominate the precision scores. In addition, while some of these false positives are "good" values misdiagnosed as anomalies by our system, it is likely that many of these errors come from suspicious values that we did not pre-flag as

existing anomalies due to a lack of domain expertise (for example, sensors 17 and 29 of Grand St. Bernard on days 22 through 25 in Figure 2.12). Thus, even though our model is catching these likely faults, each is being labeled as a false positive.

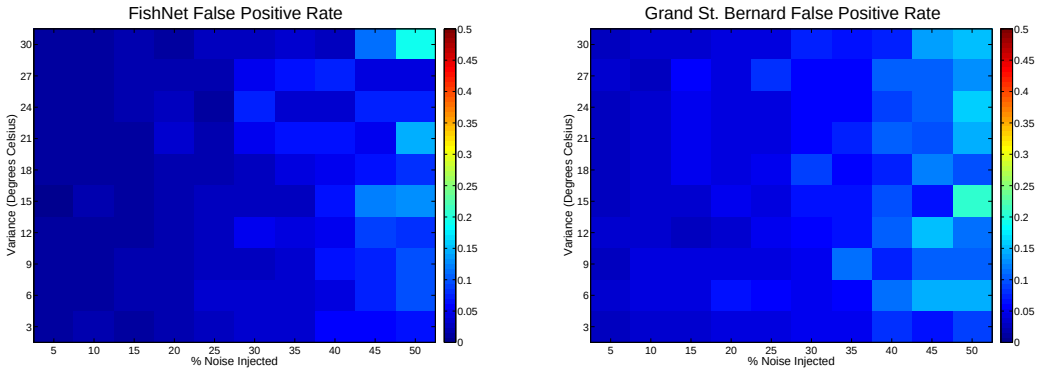

Figure 2.18: False Positive rates as a function of both noise variance (Y-axis) and percentage of data points modified by noise (X-axis). The shade of color associated with each grid cell reflects the degree of precision (on a scale of 0% to 50%) for that configuration of noise level and saturation.

There are cases where poor performance is caused by multiple sensor streams being affected by noisy values simultaneously. Consider the QC model applied to the noise-injected data in Figure 2.19 ($\eta = 50$, $\sigma_n^2 = 15$). Beginning on day 36, a cooling trend affects all stations in the deployment and reduces the true air temperature below 0 degrees ℃. Incidentally, station 29 flatlines at exactly -1.0 degrees ℃ during this period. As the true signal dips below -1.0 degrees ℃, we notice that all stations begin predicting values hovering near this boundary until the cooling trend ends around day 39. Further, instead of recognizing the values reported from station 29 as anomalous for this period (as in Figure 2.14), these values are the only ones labeled as nominal. This behavior occurs because the amount and degree of injected noised in the data makes it unlikely that a set of observations at time t will be consistent with the spatial component of the QC model, and makes it even less likely that two sets of contiguous observations (times t and $t+1$) will be consistent with the temporal transition component. During this period, station 29 behaves very consistently from a temporal perspective (its observations are constant from day 34.5 to 40.5), and the variance of the injected noise is enough to bring the observed signal from the other stations within close proximity of the flatline value. The

end result is that station 29 becomes the standard for nominal behavior until the true temperature deviates from the flatline by a margin larger than the magnitude of added noise.

In the above-mentioned scenario, increased values of both η and σ_n^2 negatively affect the precision score. As η increases, the likelihood of all sensors encountering noisy values simultaneously also rises. Noisy observations drawn from a wide-variance distribution are more likely to strongly disagree with the QC model's predictions and, as a result, this frequently causes the model to ignore such observations. This kind of scenario becomes more probable in cases where there are relatively few, highly-correlated sensors that are prone to simultaneous malfunctioning.

Recall values, displayed in Figure 2.20, are largely invariant to changing values of η. Increasing the amount of noise in the original data has no significant effect on our ability to find all data anomalies. This is unsurprising. As per our discussion of precision, an abundance of anomalous values may introduce tracking problems that leads to misclassifying normal values as anomalous; however, false positive values do not factor into recall scores. An increase in the magnitude of the noise distribution directly benefits our recall scores, which is again consistent with our hypothesis. In both data sets, we are able to achieve greater than .70 recall once the variance of the noisy data reaches 15 ℃. For the FishNet deployment, we can simultaneously reach a precision score of .87 while keeping our false positive rate at 4.6%. At Grand St. Bernard, we operate at .78 precision with a false positive rate of 7.0%. Again, Grand St. Bernard's worse performance is partially due to the presence of suspicious values that we did not preflag as data anomalies before injecting noise.

Lastly, we provide an individual analysis of the effects of increasing the frequency and magnitude of noise in each of these datasets. Figure 2.21 shows the average κ, recall, and precision for the FishNet (left) and Grand St. Bernard (right) deployments, as a function of only η (top) and only σ_n^2 (bottom). In the case of the top graphs, each value on the vertical axis represents an averaging over all values for σ_n^2; likewise, each value on the vertical axis for the bottom graphs is an average across all values for η.

Increasing the amount (η) of anomalous values in the data resulted in a increase of precision for both data sets. The effect seems less pronounced for the FishNet data set (.778 to .875) compared to Grand St. Bernard (.535 to .833) as η varies from 5% to 50%. Again, this is likely attributable to the existence of many suspicious values in the

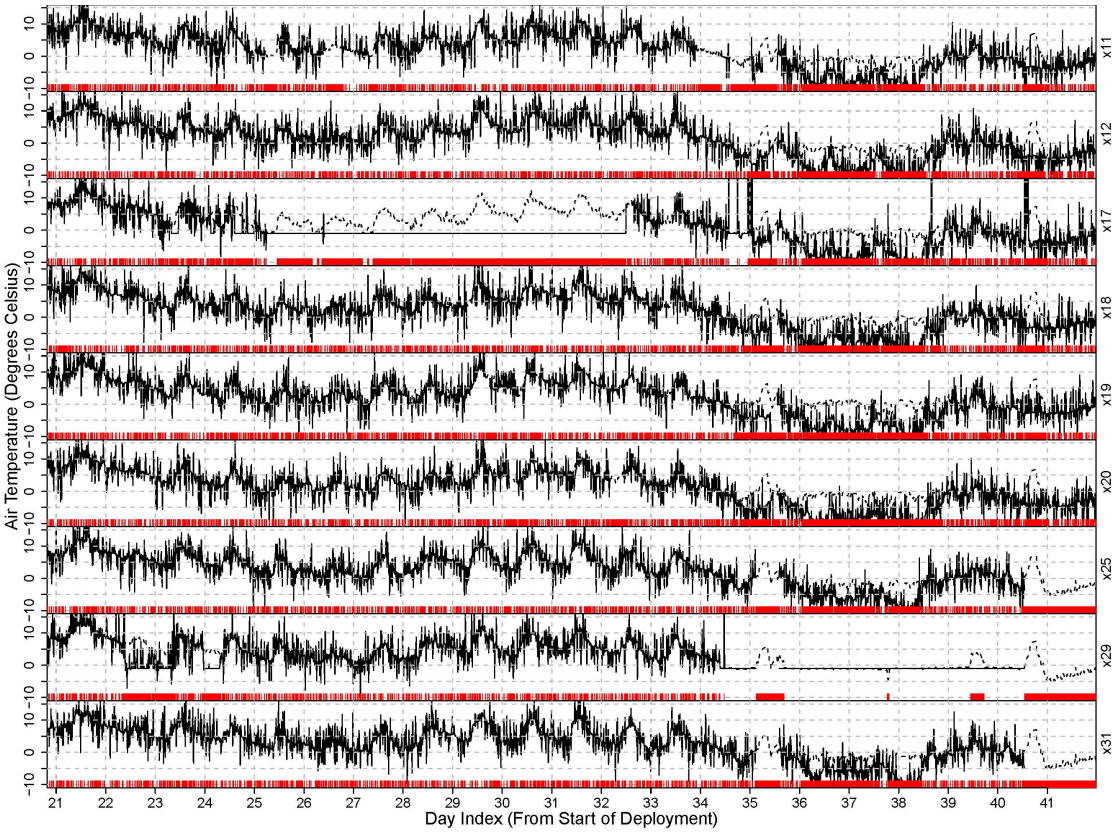

Figure 2.19: QC results for Grand St. Bernard data with 50% added Gaussian noise with variance of 15 ℃. Red hash marks depict a sensor diagnosis of *broken* for that particular value. Corresponding station names appear on the right side of the graph next to the stream they depict.

Grand St. Bernard dataset that were not preflagged as data anomalies. Prior to any Gaussian noise being added, these values would appear anomalous to our model and would be classified as data anomalies. When η is small, it is unlikely our noise injection process will target these observations and turn them into true cases of data anomalies. The end result is these observations become false positives. As η increases, more of these suspicious values are rightly cast as data anomalies during noise injection. κ increases initially with more noisy values injected into the data; however, as discussed with the κ results, we see diminishing returns and an eventual loss in κ as η grows over 25% for both the FishNet and Grand St. Bernard datasets.

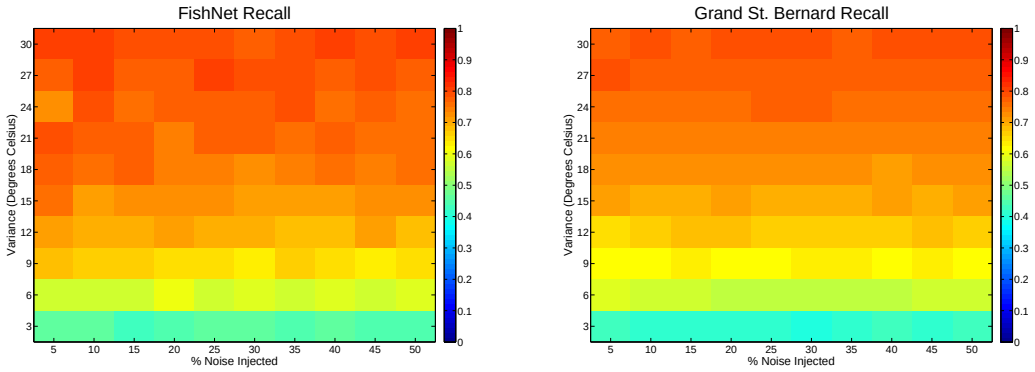

Figure 2.20: Recall as a function of both noise variance (Y-axis) and percentage of data points modified by noise (X-axis). The shade of color associated with each grid cell reflects the degree of recall (on a scale of 0% to 100%) for that configuration of noise level and saturation.

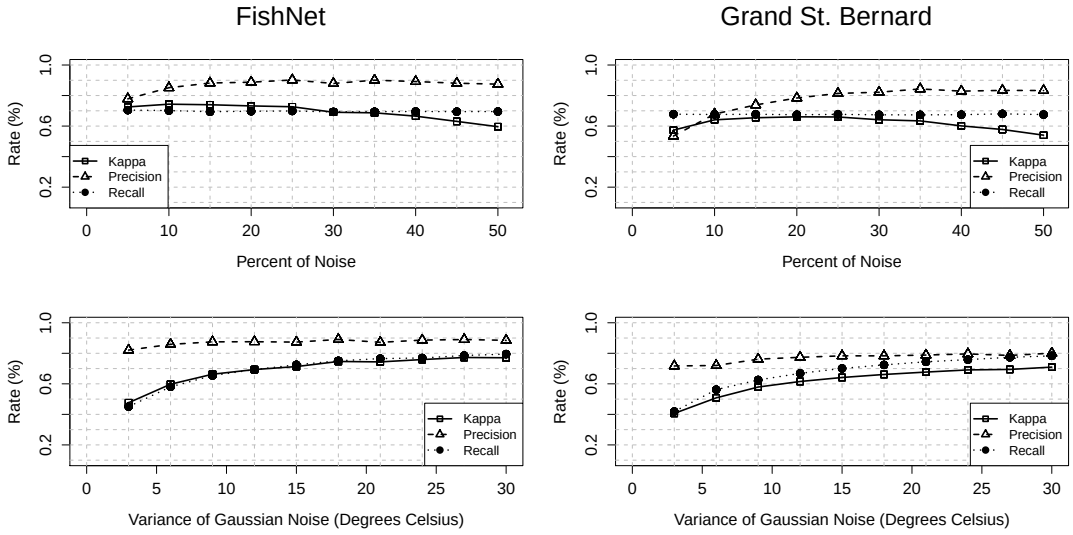

Figure 2.21: Left: Precision, Recall, and κ for the FishNet noise injection experiments as function of percentage of noise (top) and degree of variance (bottom). Right: Precision, Recall, and κ for the Grand St. Bernard noise injection experiments as function of percentage of noise (top) and degree of variance (bottom).

The lower portion of Figure 2.21 shows a definite gain in overall precision, recall, and κ as the degree of noise in the data (σ_n^2) rises from 3 ℃ to 30 ℃. Specifically, κ

increases by approximately .3 in both data sets, recall increases by .34 in FishNet and .36 at Grand St. Bernard, and precision increases by .06 at FishNet and .08 at Grand St. Bernard. In regards to recall and κ, this is because as the added noise in the data becomes more obvious (higher variance), it becomes easier for our model to detect it. The increase in precision is less pronounced than the increase in recall and κ, which could be caused by our QC model having too a great a sensitivity to anomalous values (too small of a variance for the sensor observation variable when the sensor state is believed to be *working*).

2.7.4 Noise Injection & Model Comparison

In this section we use our noise injection methodology to validate that learning a spatial model provides superior performance than arbitrarily choosing a spatial structure. We examine four spatiotemporal QC models.

- *Best*: QC model having the highest-scoring (best) spatial structure as returned from the algorithm described in Algorithm 1.
- *Worst*: QC model having the lowest-scoring (worst) spatial structure with equal connectivity (smallest vertex cut) to the Best model.
- *Full*: QC model having a spatial structure that is fully-connected ($\frac{n(n-1)}{2}$ edges among n sensors).
- *Empty*: QC model having a completely disconnected spatial model. This is identical to a temporal-only QC model.

These QC systems are compared in terms of their κ, precision, and recall scores as a function of σ_n^2 as in the bottom portion of Figure 2.21. The results can be seen in Figure 2.22.

In both the FishNet and Grand St. Bernard datasets, our learned QC Model (*Best* model) clearly outperforms the other 3 QC models in κ and precision. The difference in performance is most pronounced in the FishNet deployment. Recall scores are comparable for both FishNet and Grand St. Bernard across all levels of σ_n^2 and all four model types. The *Full* model performs slightly better than than the *Worst* model in κ and precision.

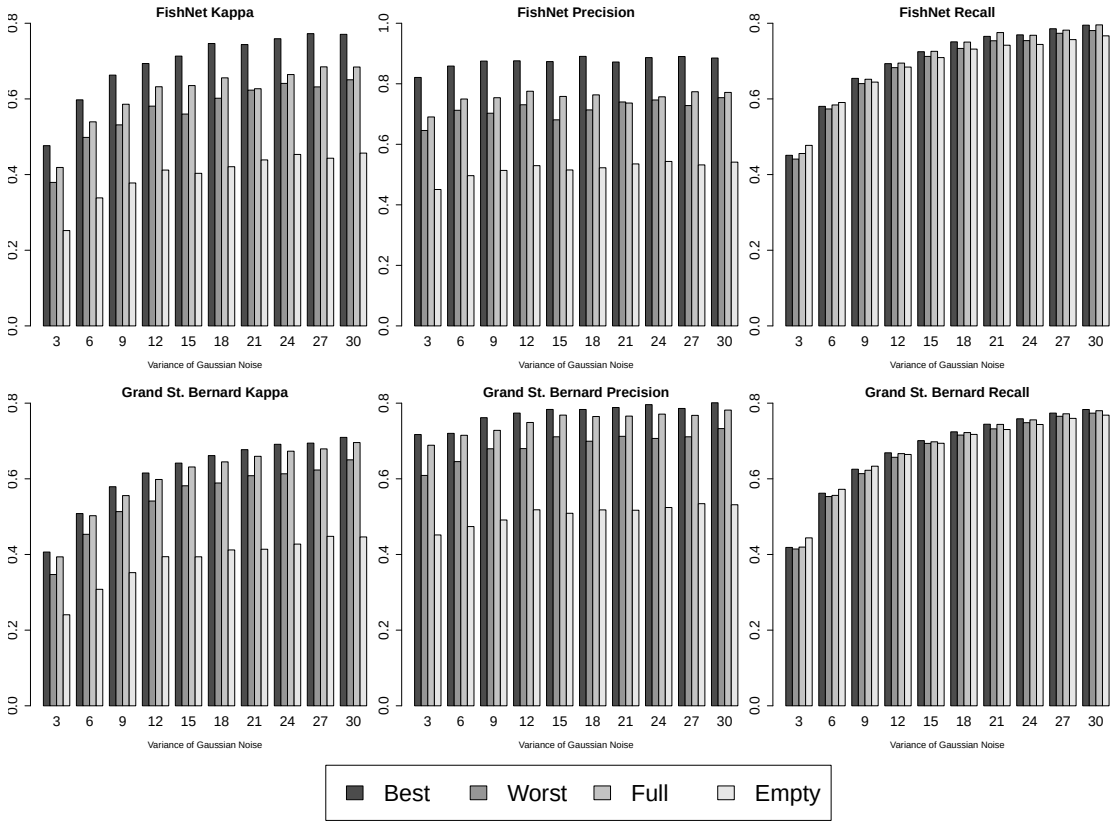

Figure 2.22: κ, Precision, and Recall for the FishNet (top) and Grand St. Bernard (bottom) noise injection experiments. X-axis refers to the magnitude of the noise injected (variance in degrees Celsius). The Y-axis corresponds to the κ, Precision, or Recall score.

This suggests that though there are disadvantages to assuming full spatial connectivity, assuming a densely connected spatial model incurs less error than assuming a spatial model with few or no connections (as in the *Empty* model). The performance gain of the *Best* QC model over the *Full* model is less significant in the Grand St. Bernard dataset. We suspect this stems from the models being trained on data observed in mid September and tested on data from mid to late October (a seasonal transition period). Both our learned spatial model and the fully-connected model will fit the test data poorly, because the training data has little resemblance to the test data. Thus, neither model has a clear advantage over the other.

2.7.5 Discussion

The statistic of principal interest to our quality control problem is recall. If our method can correctly identify and filter out all nonanomalous data points, then the expert can save time by considering only those points that our model has marked as anomalous. We want to filter out as many existing anomalies from the data prior to review by a domain expert (in order to save time) and prior to publication of the data (in order to prevent distribution of invalid measurements). Our noise injection experiments confirm that the worst case for recall is when there exist low-variance anomalies in the data. This stands to reason, for if the anomalies we need to detect fall within the range of noise for the non-anomalous data, they will be nearly-indistinguishable from the real data. Furthermore, if they are frequent, then when the model is fitted to the data, it will use them to define the normal level of variation.

Nevertheless, we have demonstrated that we can obtain recall rates of .70 when the average variance of the noise is 15 ℃ and close to .80 for values of $\sigma_n^2 \geq 20$ ℃. While this may seem unreasonable in terms of noise levels we should expect to encounter in real-world scenarios, consider that approximately 70% of the additive noise is less than or equal to one unit of standard deviation (≈ 3.87 ℃ for $\sigma_n^2 = 15$ and 4.47 ℃ for $\sigma_n^2 = 20$). In addition, we are maintaining a low false positive rate (4 to 7%) for most values of η. In cases where there are fewer complete sensor outages and the temperature data in the training distribution more closely matches that in the testing distribution, we would expect these values to further improve.

With respect to creating a sparse representation of the joint distribution of SensorScope stations, the applied structure learning algorithm resulted in a savings of parameters in both FishNet and Grand St. Bernard. Each variable (or station) in a linear-Gaussian model is specified by a scalar mean and variance (2 parameters) in addition to a weight for every parent (1 parameter for every arc in the graph). Thus, the total number of parameters for a graphical representation of the joint distribution is $2n + k$ where n is the number of variables in the model and k is the number of arcs or edges. The full joint distribution would consist of an n-dimensional mean vector and an $n \times n$ covariance matrix, of which $n + \frac{n(n-1)}{2}$ entries would need to be specified (a total of $2n + \frac{n(n-1)}{2}$ parameters). The created network structure for FishNet contained 12 arcs for a total of $12 + 2 * 6 = 24$ parameters. The regular full joint distribution would re-

quired $2 * 6 + \frac{6(6-1)}{2} = 27$ parameters. The savings in this case was minimal; however, the FishNet station covers a relatively small and homogeneous area compared to Grand St. Bernard and thus we expect some measurable correlation among all the sensors at the deployment. The Grand St. Bernard network consisted of 24 edges, so the Bayesian network representing this model requires $2 \times 9 + 24 = 42$ parameters. The full joint distribution would require $2 * 9 + \frac{9(9-1)}{2} = 54$ parameters to be completely specified. We see that the savings, in terms of parameters to be estimated, grows very quickly as the number of sensors increases, and that we can exploit spatial 1heterogeneity to provide a more compact representation. The number of spatial structures we considered in determining the best spatial models (size of the MEC for B_{post} in Table 2.6.4) for FishNet and Grand St. Bernard were 3 and 8, respectively. This result is consistent with empirical data obtained by experiments involving the evaluation of MEC class sizes [26].

Finally, our empirical results also show that sparseness in the spatial model represents a form of regularization. Figure 2.22 shows that fully connected spatial models behave worse or on par with our learned spatial models, which contain fewer arcs. As we cannot evaluate the entire set of all spatial models for each dataset, we cannot be certain there do not exist even sparser models that perform better. However, we have some evidence from the *Worst* and *Empty* models' performance in Section 2.7.4 that we cannot capture all the necessary correlative relationships with fewer edges. The *Worst* spatial model for FishNet contained 10 edges compared to 12 in the learned model, and the *Worst* spatial structure for Grand St. Bernard contained 19 arcs compared to 24 in the learned model; neither performed as well as the *Best* model in each dataset. The placement of the edges in the *Worst* model is an additional contributing factor to its poor performance.

2.8 Related Work

A simple (though common) approach to data-anomaly detection is to provide a visual representation of the data and allow a domain expert to manually inspect, label, and remove anomalies. In Mourand and Bertrand-Krajewski [49], this method is improved upon through the application of a series of logical tests to pre-screen the data. These tests include range-checks to insure that the observations fall within reasonable domain limits, similar checks for the signal's gradient, and direct comparisons to redundant sensors. The ultimate goal is to reduce the amount of work the domain expert has to do to clean the

data, which is consistent with our approach.

Temporal methods evaluate a single observation in the context of a time segment (sliding window) of the data stream or previous observations corresponding to similar periods in cyclical time-series data. Reis et al. [60] use a predictor for daily hospital visits based on multiday filters (linear, uniform, exponential) that lend varying weight to days in the current sliding window. The motivation for such an approach is to reduce the effect of isolated noisy events creating false positives or false negatives in the system, as might occur with a single-observation-based classifier. In a similar vein, Wang et al. [69] construct a periodic autoregressive model (PAR, [7]), which varies the weights of a standard autoregressive model according to a set of user-defined periods within the time series. A daily visitation count is predicted by the PAR model, and if it matches the observed value, then the PAR model is updated with the observation; otherwise, the value is flagged as anomalous, an alarm is raised, and the observation is replaced with a uniformly smoothed value over a window containing the last several observations. A machine-learning based approach was adopted by Wong et al. [37] wherein the logical tests, or rules, are learned in an online setting. Past observations (taken from set lag periods representative of current temporal context) are mined for rules stated as reasonable values for individual, pairs, or tuples of attributes. The significance of the rules are determined by Fisher's Exact Test.

Spatial methods are useful in cases where there exist additional sensors distributed over a geographic area. The intuition is that if an explicit spatial model exists that can account for the discrepancies between observed values at different sites, then these sensors can, in effect, be considered redundant. An example of this approach can be found in Daly et al. [12], where each distributed sensor is held out from the remaining set of sensors, and its recorded observation validated against an interpolated value from the remaining set. Each station's value in the network is given a weight associated with confidence in its estimate. This confidence value is calculated using a set of summary statistics based on that station's latest observation in the context of its historical record. Unlike our approach, there is no specific attempt to model the joint distribution between all stations or the overall correlation between sensors in the network. Moreover, this approach relies on a significant historical record for each station in the network in order to compute the necessary summary statistics for that station.

Belief Networks [54] have been employed for sensor validation and fault detection in

domains such as robotic movement, chemical reactors, and power plant monitoring [52, 46, 32]. Typically, the uncertainty in these domains is limited to the sensor's functionality under normal and inoperative conditions. That is, the processes in these domains function within some specified boundaries with a behavior that can be modeled by a system of known equations [33, 3]. Ecological domains are challenging because accurate process models encompassing all relevant factors are typically unavailable [28]; consequently, uncertainty must be incorporated into both the process and sensor models. Eskin [21] handles this uncertainty with a mixture model over the true and anomalous data, which is similar to our observation variable once we have marginalized away the sensor state. The distribution parameters are learned iteratively over each sample in the dataset. For each value, the change in likelihood of moving that value's membership from the clean to the anomalous distribution is computed; if the the likelihood increases, the value changes membership, it becomes anomalous permanently (it cannot rejoin the clean distribution), and the nonanomalous distribution parameters are updated. Das et al. [13] use the probabilistic approach in the multivariate setting in which rare co-occurrences of attribute values are not, in and of themselves, indicative of anomalous values. Here, pairs or tuples of attributes are probabilistically scored based on their values; however, they are normalized by likelihood of the individual values taking on those assignments independently, as determined by the training data (to add support to low-frequency events). An entire record (consisting of multiple attribute tuples) is then scored according to the rarest tuple of attribute values within that record.

Perhaps most related to our own work, Hill et al. [29] apply a DBN model to analyze and diagnose anomalous wind velocity data. The authors explore individual sensor models as well as a coupled-DBN model that attempts to model the joint distribution of two sensors. The nature of the data-anomaly types in the data appear to be either short-term or long-term malfunctions in which the wind speed drastically increases or decreases; consequently, a first-order Markov process is sufficient to determine sharp rates of increase or decrease in wind speed. The joint distribution is modeled as a multivariate Gaussian conditioned on the joint state of the respective sensors (represented as a discrete set of state pairs). Our current approach primarily differs in the scale (number of sensors we are trying to simultaneously monitor) and that we attempt to discover the correlative structure between the sensors. Instead of assuming a full covariance matrix over the joint distribution of sensors and computing the MLE parameters for that matrix,

we apply structure learning to obtain a sparse representation of the joint distribution.

2.9 Concluding Remarks and Notes on Future Research

This article has described a new type of dynamic environmental monitoring based on short-term wireless sensor deployments, as well as demonstrated an accompanying need for adaptive, automated quality control. We have provided background information regarding the SensorScope Project and have given examples of the data collected and the data anomalies contained therein. However, our primary contribution has been to offer an initial means of automating QC in this domain. Our experimental results thus far demonstrate that a Dynamic Bayesian Network approach, based on a generative model of the deployment site, can diagnose many of the data-anomaly types present in ecological data. Further, in all but the severest case of a complete site outage, the model is able to reconstruct reasonable estimates of missing or corrupted data from individual sensors or subsets of sensors. We have shown that structure learning techniques can be successfully applied in this domain to learn a compact representation of the covariance matrix over the generative distribution, and that this sparse matrix performs better or comparable to a fully specified covariance structure.

Thus far we have only applied our method to detect data anomalies present in air temperature sensor streams. We suspect that other environmental data types may provide more challenges to our approach. For example, wind velocity sensors may demonstrate significantly less temporal and spatial correlation over the relatively small geographical areas they are deployed, or surface-temperature data may not be very spatially correlated if the observation area displays surface heterogeneity. However, a model that examines the correlation across these phenomenon may overcome these challenges. Other domains that may be difficult to perform QC on exclusively (precipitation, soil moisture, solar radiation) may be leveraged with other correlated phenomena to produce a truly inclusive system for quality control.

With regards to structure learning, the BGe metric and hill-climbing search represent only one prior-based technique for determining the underlying spatial model. In learning a compact form of the covariance matrix, there appear to be two primary methodologies. Standard machine learning approaches focus on the discovery of some metric to score child/parent configurations and a search algorithm over the space of DAGs that

may penalize nonsparse representations. For example, Tsamardino et al. [67] attempt to determine a candidate set of neighbors (the Markov Blanket) using a Max-Min Parents heuristic for each variable in the network and then employ hill-climbing over the subset. Schmidt et al. [62] similarly attempt to identify a subset of variables to consider as a potential set of neighbors using L1-Regularization. However, both of these methods were developed to address very large databases containing thousands of variables (gene expression data, etc.) with relatively few samples – situations in which overfitting is a significant concern. While ecological sensor networks may include dozens of sensors at a deployment, it seems unlikely that the aforementioned techniques would be necessary due to the quantity and frequency at which data is collected. The second approach focuses on learning the covariance matrix directly rather than iteratively. While perhaps not feasible for domains of high dimension, this method does have the advantage of performing a more global evaluation of the structure, making it more robust to local maxima (unlike hill-climbing). The work of Yuan and Li [72] is an example of this approach, where Lasso regression is used as part of an optimization to force off-diagonal elements of the covariance matrix toward 0. The result is an undirected graph representing the covariance structure and, as we are not primarily interested in developing causal models of the sensor correlations, this would be suitable in the current domain. Also of interest would be a way to integrate our fixed temporal model into the hill-climbing search for an optimal spatial structure. That is, we would like the scoring function to take into account that lag variables will be appended to each of variables in the graph in a specific manner when evaluating candidate structures.

An additional direction for future work is to extend this model to an online learning scheme, in which the spatial structure and parameterization is refined over time. Given that the BGe metric requires a prior network structure as an initialization point for search, one could conceive of an algorithm in which the network learned on incremental batches of observations served as the prior for the next network. We could begin with a very weak assumption on the generative distribution (total independence among sensors with each sensor having a univariate Normal distribution over the range of plausible domain values), and use this as our initial QC system. Those points not labeled as anomalous by this primitive model would then be employed to train a more sophisticated spatial model, and then the process could be repeated. On a related note, there may be other metrics for conditional independence that merit exploration, especially if we are to

loosen our assumption on normally distributed generative model or on a linear correlative relationship between the variables.

2.10 Acknowledgments

We would like to thank Marc Parlange and the EFLUM Group for hosting and supporting this research at the EPFL. We would also like to thank Mounir Krichane and Guillermo Barrenetxea for providing us with the SensorScope datasets and for their technical advice regarding the sensor stations. Finally, we wish to acknowledge Marc-Olivier Boldi for his statistical expertise and his contributions to the development and debugging of our QC model. This work was supported under the National Science Foundation (NSF) Ecosystem Informatics IGERT (grant number DGE-0333257) and funding from the EPFL.

Chapter 3: Manuscript Two

Scalable Inference for Probabilistic QC

Ethan W. Dereszynski and Thomas G. Dietterich

To be submitted.
May 2012

3.1 Abstract

Machine-learning approaches to automated quality control have demonstrated great potential for assisting ecological scientists in cleaning data collected from environmental sensor networks. We motivate a probabilistic approach that captures the uncertainty in the working state of sensors and the value of the phenomenon they measure. The probabilistic model contains both discrete and continuous random variables, which makes inference intractable for modern, larger sensor networks that may have dozens or hundreds of sensors. We examine the performance of three approximate methods for inference in this domain. Two of these algorithms represent contemporary approaches to inference in hybrid models, while the third is a greedy search-based method of our own design. We demonstrate the results of these algorithms on synthetic datasets and a real environmental sensor data gathered from an ecological sensor network located in western Oregon. Our results suggest that we can improve performance over networks with less sensors that use exhaustive asynchronic inference by including additional sensors and employing approximate techniques. The greedy search algorithm achieves the best results for a network of 27 sensors that includes measurements for air temperature, mean wind, solar radiation, and precipitation.

3.2 Introduction

Remote sensors have become an invaluable tool for monitoring environmental phenomena across multiple temporal and spatial scales. Advances in sensor technology have not only decreased costs and improved availability, they have widened the scope of what automated sensors can capture. In addition to sensors for measuring standard environmental variables (air temperature, precipitation, etc.), scientists are now deploying bioacoustic sensors to monitor bird species, flux towers to detect changes in CO_2 in both soil and air, and image capture for species identification ([58]). Dozens to hundreds of these sensors can be distributed throughout a landscape, providing an unprecedented view of the complex ecological processes at work within it [5, 57, 9].

Ecological research organizations, such as the Long Term Ecological Research (LTER) network and National Ecological Observatory Network (NEON), have readily adopted these technologies to create continental-scale sensor networks. A goal of these agencies is

to identify the drivers behind, and validate existing models of, ecological processes occurring at different spatiotemporal scales using measurements gathered from instrumented sites. However, before such a task can be undertaken, sensor data must be quality controlled (QC'd) to remove invalid readings caused by sensor failure. This is particularly relevant to environmental sensor data, as the in-situ nature of the sensors makes them prone to malfunction. Such malfunctions are exhibited by biased readings, calibration errors, and signal loss (to name a few). As model forecasts and analyses of the collected data may be used to guide future ecological research, and ultimately influence policy decisions, it is paramount that the data undergo QC prior to its integration.

Here, we consider quality control of environmental sensor data. Data from a sensor network is treated as a multivariate time series, with each dimension of the series corresponding to measurements taken by a single sensor at a fixed sampling frequency. Given such a dataset, an ideal quality control scheme should demonstrate three qualities:

- The approach must flag observations that are likely to be corrupted by sensor failure.
- It should support a means of gap filling by providing a best-guess imputation of the affected observation.
- The scheme should be generalizable to different types of sensor data, deployment locations, and fault types.

Related to the first two properties, a confidence measure in the classification of an observation being "good" or a "data anomaly," as well as the imputation of affected values, should also be provided. The confidence measure allows data consumers to determine what values are acceptable for their purposes.

Many of the approaches used by information managers to perform QC typically fall short in one of the aforementioned objectives. For example, a common method is to apply a series of range checks and remove measurements that fall outside of acceptable levels (for example, air temperature measurements > 50 ℃). While fast to apply to large volumes of data, these methods fail to detect data-anomalies that may occur within reasonable limits, nor can they distinguish values on the edge of these limits (questionable measurements) from those in the middle (nominal measurements) [64]. After initial range checks, a common follow-up is manual inspection of the plotted time series by a domain

expert. Experts can identify abnormal behavior of a sensor based on their knowledge of the hardware and the phenomenon it is measuring. Unfortunately, such inspection is infeasible for dozens or hundreds of sensors recording at very fine temporal resolutions. Neither range checks nor visual inspection provides a solution for gap-filling in place of affected values.

In this paper, we pursue a machine learning-based approach, wherein we learn a probabilistic model $P(X)$ of the latent process generating observations at each sensor. This process model is learned directly from the sensor data. Uncertainty in the working state of the sensor is encoded in a probability distribution over its state $P(S)$. A sensor model $P(O|X, S)$ couples sensor observations O to the process model according to the state of the sensor. In effect, we treat sensor measurements as noisy observations of this latent process, where the degree of noise is linked to the functioning state of the sensor; i.e., whether the sensor is working or broken. Queries regarding the working state of the sensor (and the quality of its readings) are resolved through statistical inference. An advantage of the probabilistic framework is that it provides a natural interpretation of "confidence" in the form of probability values. The data-driven learning of the process model minimizes the amount of domain expert knowledge required to configure this method to a new site. Moreover, by avoiding modeling specific ways in which a sensor can fail, the model can generalize to new malfunctions that may arise over the course of a deployment.

In previous work [18], we developed and tested this probabilistic approach for deployments involving fewer than 10 sensors. The purpose of this paper is to address two shortcomings of that previous work. First, the algorithms that we developed for probabilistic inference scale exponentially in the number of sensors, which makes them infeasible for large sensor networks. In this paper, we study methods that scale much more practically. Second, the methods developed in the previous papers considered only a single kind of sensor (a thermometer). In this paper, we evaluate our algorithms on networks of heterogeneous sensors.

The remainder of the article is outlined as follows. In Section 3.3, we provide a formal definition of our probabilistic QC model, and describe the inference queries (and their associated costs) posed to the model. Next, we outline the three approximate inference algorithms used in this work: Rao-Blackwellized particle filtering (RBPF), Expectation Propagation (EP), and SearchMAP, our own greedy-based method for approximate infer-

ence. Then, we introduce the real datasets used for our experiments, the different types of sensor data being analyzed, and our methodology for generating synthetic records. Section 3.6 explains the experimental set up used to analyze each of the models, including results and relevant discussion. Lastly, we conclude with related work being done in quality control, specifically as it relates to machine learning-based approaches.

3.3 A Probabilistic Model for Quality Control

Our quality control method employs a probabilistic approach to represent two key sources of uncertainty in the environmental monitoring domain. The first source comes from the processes we are trying to directly monitor through sensors (i.e., air temperature at multiple locations, solar radiation, etc.). Even when a sensor is operating correctly, physical limitations in the measurement device inject some inaccuracy into each recorded value. Consequently, we assume observations from each sensor are, in the *working* case, noisy observations of this latent process and, in the case where the sensor is *broken*, uncorrelated noise. The second source is the operational state of sensor itself. In many cases of sensor failure, the sensor will continue to collect and transmit observations after the malfunction has occurred, making the value of the affected data the only indicator of its working or broken state. In this section, we present a dynamic Bayesian network (DBN) for modeling these two sources of these uncertainty [14]. We also provide a description of the inference queries used in this model to reason about our uncertainty.

3.3.1 The DBN Model

We begin by considering a deployment consisting of a single environmental sensor. The sensor provides a scalar observation of some underlying ecological process X^t at time t, where t is a fixed time interval determined by the sensor's sampling frequency. We assume that the value x_t of the latent process at each sampling interval is drawn from a Gaussian distribution:

$$x_t \sim P(X^t) = N(x^t; \mu_x, \sigma_x^2) = (2\pi\sigma_x^2)^{-\frac{1}{2}} \exp \frac{\left(x^t - \mu_x\right)^2}{2\sigma_x^2}.$$

We refer to this distribution, $P(\mathbf{X})$, as the process model. As stated earlier, observations from the sensor, denoted $O^t = o^t$, are dependent on the value of this process, as well as the working state of the sensor S^t at time t. More formally, the distribution of O^t is conditioned on both X^t and S^t, such that

$$P(O^t = o^t | X^t, S^t) = \begin{cases} N(o^t; X^t, \sigma_w^2) & \text{if } S^t = 0 \text{ (sensor is working)} \\ N(o^t; 0, \sigma_b^2) & \text{if } S^t = 1 \text{ (sensor is broken),} \end{cases}$$

where σ_w^2 is some small additional noise term when the sensor is *working*. We choose σ_b^2 to be large, so that when the sensor is *broken*, $P(O^t | S^t = broken)$ approximates a Uniform distribution over all possible observation values (i.e., if the senor is broken, then all observations are equally probable). We represent the discrete-valued sensor state S^t using a Bernoulli distribution: $P(S^t = working) = p$ and $P(S^t = broken) = 1 - p$. We refer to $P(\mathbf{O}|\mathbf{X}, \mathbf{S})$ and $P(\mathbf{S})$, together, as the observation model. The advantage of this non-specific fault model is that we do not explicitly have to enumerate different types of sensor failure, which allows our approach to generalize to new types of sensor failure unseen in previous sensor data.

The assumption of a fixed mean parameter for $P(\mathbf{X})$ is often inappropriate for environmental data. For example, seasonal and diel trends effects can cause the observed air temperature, soil temperature, and solar radiation (to name a few) to fluctuate over time. To account for this, we include a temporal component into the process model, such that X^t is conditioned on X^{t-1}. In this way, the state of the process model at time $t-1$ can be used to influence our belief about the process at time t. This influence is defined by a first-order Markov model, in which the mean of $P(X^t)$ is a linear function of X^{t-1}

$$P(X^t = x_t | X^{t-1} = x^{t-1}) = N(x_t; \mu_x + w_x x^{t-1}, \sigma_x^2).$$

The single-sensor model has a significant drawback—observations at the sensor provide the only source of current information about the latent process. If the sensor state becomes *broken*, our uncertainty about the state of the process will grow larger each time step, and we will have no way to impute what the sensor "should" have observed. To understand why this occurs, consider how the variance of $P(X^t)$ grows after we marginalize

away our uncertainty about X^{t-1}:

$$\int_{X^{t-1}} P(X^t|X^{t-1})P(X^{t-1})dX^{t-1} = N(\mu_x + w_x\mu_{x^{t-1}}, \sigma_x^2 + w_x^2\sigma_{x^{t-1}}^2).$$

It is for this reason that it is common practice to deploy multiple sensors at a site. As the price of sensors continues to drop, we expect this practice will become even more common. Consequently, as long as the number of simultaneously-failing sensors is small, we can exploit redundancy and correlations among the sensors to detect anomalies and impute corrected values.

To that end, we can extend the single-sensor model to represent the joint relationship among N sensors at a deployment by instantiating a separate X_i^t, O_i^t, and S_i^t for each of the $i = 1, \ldots, N$ sensors. Let $\mathbf{X}^{t-1} = \left\{X_1^{t-1}, \ldots, X_N^{t-1}\right\}$ and $\mathbf{X}^t = \left\{X_1^t, \ldots, X_N^t\right\}$, then the process model $P(\mathbf{X}) = P(\mathbf{X}^{t-1}, \mathbf{X}^t)$ takes on a multivariate Normal density,

$$\begin{aligned} P(\mathbf{X}^{t-1}, \mathbf{X}^t) &= MVN(\vec{x}^{\ t}; \vec{\mu}_X, \Sigma_X) \\ &= \frac{1}{(2\pi)^{N/2}|\Sigma_X|^{1/2}} \exp\left[-\frac{1}{2}(\vec{x}^{\ t} - \vec{\mu}_X)^T\Sigma_X^{-1}(\vec{x}^{\ t} - \vec{\mu}_X)^T\right]. \end{aligned}$$

In previous work, we described a method for learning a linear-Gaussian Bayesian network that encodes the parameters of the process model: $\vec{\mu_X}$ and Σ_X [18]. Following that approach, each variable in $P(\mathbf{X})$ is represented by a node in a directed acyclic graph (DAG), and edges in the graph indicate conditional dependencies among the variables. Each variable X_i^t is distributed as Gaussian, with a mean that is a linear function of its parents (the set of nodes with a directed edge pointing to X_i^t, denoted $Par(X_i^t)$),

$$P(X_t^i|Par(X_t^i)) = N(\mu_i + \sum_{j \in Par(X_t^i)} w_j * X_j, \sigma_i^2).$$

The joint distribution $P(\mathbf{X}^{t-1}, \mathbf{X}^t) = \prod_{i=1}^N P(X_i^t|Par(X_i^t))P(X_i^{t-1}|Par(X_i^{t-1}))$ also follows a multivariate Normal density. The structure of the linear-Gaussian network is learned via a hill-climbing search algorithm and the BGe scoring metric [24]. The BGe metric combines a provided Bayesian network (posited as a prior over network structures) with a set of training data and arrives at a scoring function that evaluates candidate networks based on the likelihood of being generated by a posterior Normal-Wishart

distribution. The Normal-Wishart distribution can be thought of as a distribution over mean vectors and covariance matrices, each of which can be encoded in a linear-Gaussian Bayesian network. A result of our approach was that the learned process models often generalized better to new sensor observations than those that used the maximum-likelihood estimators (MLE) of $P(\mathbf{X})$. The ability of the model to generalize to new test data was noted by its superior performance in the QC task, in addition to other metrics (we refer the reader to that work for more details of this algorithm and experimental results).

Finally, our single-sensor observation model is replicated for each of the N sensors in the deployment, with each sensor's observation O_i^t conditioned on its respective latent process variable X_i^t and sensor state variable S_i^t. The complete model is shown for two time steps in Figure 3.1. The model is a dynamic Bayesian network, where each slice of the network contains a conditional-linear Gaussian (CLG) structure. Note that we do not show the internal structure among the variables in the process model, as this component is learned separately at each deployment.

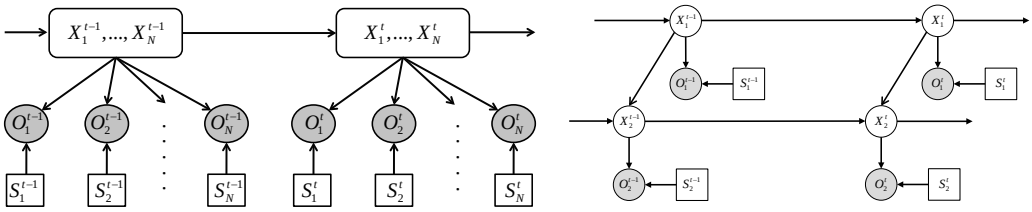

Figure 3.1: Left: Abstract representation of two time slices of the dynamic Bayesian network model used for quality control. Square nodes denoted Bernoulli-distributed variables, and circular nodes denote Gaussian-distributed variables. Shaded variables indicate the sensor observations; those variables that we can directly observe at each time slice. Right: Two time slices of an example QC model for a network with two sensors.

3.3.2 Inference for Quality Control

At each time slice, we are interested in the results of two probabilistic queries. The first query determines the assignment of $\{working, broken\}$ to each of the sensor state variables that best explains the set of observations $\vec{o}^{\,t} = \{o_1{}^t, \ldots, o_N{}^t\}$. We refer to this

query as the *maximum aposteriori*, or MAP, query of the sensor states:

$$\underset{S^t}{\operatorname{argmax}} P(S_1^t, \ldots, S_N^t | O_1^t = o_1^t, \ldots, O_N^t = o_N^t). \tag{3.1}$$

If the MAP value of a sensor state variable $S_i^t = \mathit{broken}$, this means that the most likely interpretation of the observation $O_i^t = o_i^t$ (under our model) is that it is a data-anomaly. Therefor, the solution to this query meets our first requirement for a QC method: it provides flags for those values that are indicative of sensor failure (specific to the sensor that observed it) at each time step. This type of query is sometimes referred to as a "MAP-marginal" query, because it requires that uncertainty about the process variables $\mathbf{X}$ be marginalized away before determining the most likely configuration of sensors [38].

The second query computes our updated belief about the latent process at time t given observations from each of the sensors up to that time:

$$P(X_1^t, \ldots, X_N^t | O_1^t = o_1^t, \ldots, O_N^t = o_N^t). \tag{3.2}$$

This query serves two purposes. First, we can use it to obtain an imputation of the missing "true" value for observations we flag as data-anomalies, along with a measure of confidence in that imputation. We can supply additional evidence in the form of a specific sensor-state configuration $\vec{s}^{\,t} = \left\{s_1^t, \ldots, s_N^t\right\}$ (computed in 3.1) to the query $P(\mathbf{X^t} | \mathbf{O}^t = \vec{o}^{\,t}, \mathbf{S}^t = \vec{s}^{\,t})$ to yield a posterior distribution over the value of the latent process. The mean vector of this distribution can be interpreted as a point estimate of the current value of the latent state. Specifically, we can use μ_i^t as an estimate of what the observation o_i^t should have been if we believe $S_i^t = \mathit{broken}$, and the covariance entry $\Sigma_{i,i}^t$ measures our uncertainty in that estimate. By introducing the sensor states as additional evidence, we insure that the imputation is consistent with decision to declare a sensor *broken* in a time step. Second, recall that our process model's belief about the current time step is conditioned on its belief in the previous time step. Hence, we also need the joint distribution in (3.2) to represent our past belief about $P(\mathbf{X}^t)$ when we repeat queries (3.1) and (3.2) at time $t + 1$.

Unfortunately, the computational cost of both queries (3.1) and (3.2) scale exponentially with N, the number of sensors in the network. To provide some intuition for this scaling cost, consider a single slice of our QC model having $N = 3$ sensors and ignoring

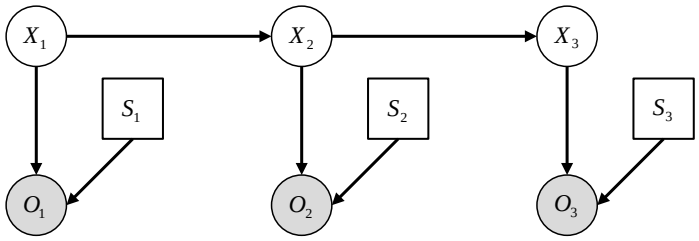

Figure 3.2: A single-slice example of graphical model used in our QC method. This example includes 3 sensors. Temporal arcs linking X_1, X_2 and X_3 to themselves at times $t-1$ and $t+1$ are discarded to demonstrate *asynchronic* (within a single time step) inference costs.

$P(\mathbf{X})$'s temporal component, as shown in Figure 3.2. To evaluate the MAP query, we compute

$$\begin{aligned}\operatorname*{argmax}_S P(\vec{o}|\mathbf{S}) &= \operatorname*{argmax}_{S_1,S_2,S_3} \int_{X_1}\int_{X_2}\int_{X_3} P(S_1)P(X_1)P(o_1|X_1,S_1)P(S_2)P(X_2|X_1) \\ &\quad P(o_2|X_2,S_2)P(S_3)P(X_3|X_2)P(o_3|X_3,S_3)dX_3dX_2dX_1 \\ &= \operatorname*{argmax}_{S_1,S_2} \int_{X_1}\int_{X_2} P(S_1)P(X_1)P(o_1|X_1,S_1)P(S_2)P(X_2|X_1) \\ &\quad P(o_2|X_2,S_2)\operatorname*{argmax}_{S_3}\int_{X_3} P(S_3)P(X_3|X_2)P(o_3|X_3,S_3)dX_3dX_2dX_1.\end{aligned}$$

The result of the multiplication and integration of the terms inside the innermost argmax is a Gaussian potential over S_3 and X_2; it has a Gaussian mixture density with 2 components, as shown in Figure 3.3 (left). Let us denote this potential as $\Phi(S_3, X_2)$. Then, in the next step of the computation:

$$\begin{aligned}\operatorname*{argmax}_S P(\vec{o}|\mathbf{S}) = &\operatorname*{argmax}_{S_1}\int_{X_1} P(S_1)P(X_1)P(o_1|X_1,S_1) \\ &\operatorname*{argmax}_{S_2}\int_{X_2} P(S_2)P(X_2|X_1)P(o_2|X_2,S_2)\Phi(S_3,X_2)dX_2dX_1.\end{aligned}$$

Now, we have the product of two Gaussian potentials inside the innermost argmax: $\Phi(S_2, X_1)\Phi(S_3, X_2)$. The result of this product $\Phi(X_1, S_2, S_3)$ is also a mixture of Gaussians, but now with 4 components, each corresponding to a configuration of $\{working,$

broken} to S_3 and S_2. In general, after N steps in this computation, we will have an interim potential Φ that contains a mixture of 2^N Gaussians.

A similar problem arises when computing query (3.2). Here, we want to compute the joint posterior distribution over the latent process given observations up to the current time step. We accomplish this by marginalizing away our uncertainty about each sensor-state variable S_i.

$$\begin{aligned}
P(\mathbf{X}|\mathbf{O}=\vec{o}) &= \sum_{S_3}\sum_{S_2}\sum_{S_1} P(S_3)P(X_3|X_2)P(o_3|X_3,S_3)P(S_2)P(X_2|X_1) \\
&\quad P(o_2|X_2,S_2)P(S_1)P(X_1)P(o_1|X_1,S_1) \\
&= \sum_{S_3}\sum_{S_2} P(S_3)P(X_3|X_2)P(o_3|X_3,S_3)P(S_2)P(X_2|X_1) \\
&\quad P(o_2|X_2,S_2)\sum S_1 P(S_1)P(X_1)P(o_1|X_1,S_1) \\
&= \sum_{S_3} P(S_3)P(X_3|X_2)P(o_3|X_3,S_3)\sum_{S_2} P(S_2)P(X_2|X_1) \\
&\quad P(o_2|X_2,S_2)\Psi(X_1)
\end{aligned}$$

The conditional distribution of the latent process given the state of the sensor and its observation $P(X_i|S_i, O_i = o_i)$ is shown in Figure 3.3 (left). In the last line of the above computation, we marginalize out the sensor state variable S_1 from this distribution, which results in a Gaussian mixture $\Psi(X_1)$, shown in Figure 3.3 (right, solid line). However, the mixture has the same number of parameters as the original conditional distribution:

$$\Psi(X_1) = pN(\mu_1, \sigma_1^2) + (1-p)N(\mu_2, \sigma_2^2).$$

Maintaining the true form of each mixture for the remainder of the computation would result in a final Gaussian mixture $\Psi(X_1, X_2, X_3)$ that would have $2^3 = 8$ components. When computing the state of the latent process at $t+1$,

$$P(\mathbf{X}^{t+1}|\mathbf{O}^{1:t+1}) = P(\mathbf{X}^t|\mathbf{O}^{1:t})P(\mathbf{X}^{t+1}|\mathbf{X}^t)P(\mathbf{X}^{t+1}|\mathbf{O}^{t+1}),$$

note that we are multiplying a 2^N-component mixture of Gaussians from t (term 1) with a new 2^N mixture of Gaussians computed at $t+1$ (term 3). The result is a 2^{2N}-component Gaussian mixture. More generally, we see that query (3.2) scales exponentially with time

and the number of sensors, requiring 2^{Nt} to compute exactly at time t.

An alternative to maintaining an exponentially increasing Gaussian mixture is to approximate the query by collapsing this mixture into a single component whenever we marginalize away a sensor-state variable (Figure 3.3, right, dashed line). This collapsing is done so as to obtain the best representation of a 2-component mixture of Gaussians with a single component, as measured by KL-divergence [41]. The equations for the collapsed mean and variance are calculated as

$$\mu_c = \sum_{k=1}^{K} p_k \mu_k \tag{3.3}$$

$$\sigma_c^2 = \sum_{k=1}^{K} p_k \sigma_k^2 + \sum_{k=1}^{K} p_k (\mu_k - \mu)^2, \tag{3.4}$$

where K is the number of components (here, $K = 2$), and p_k, μ_k, and σ_k^2 are the weight, mean, and variance, respectively, of the k^{th} component.

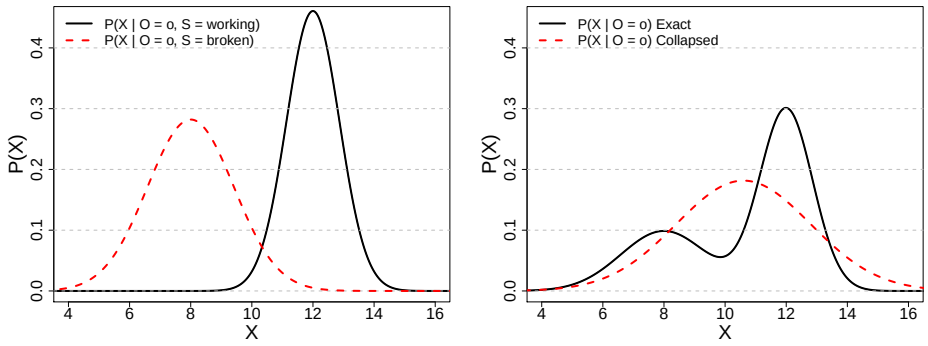

Figure 3.3: Left: The conditional distribution on $P(X|S, O = o)$. Right: After marginalizing our uncertainty about the sensor state, the exact distribution is a mixture of Gaussians (solid black line). We can collapse this Gaussian onto a single component (red dashed line) to reduce the number of parameters needed to represent this distribution.

Lerner and Parr showed that the exponential cost (in the number of discrete variables) of exact inference is inherent to any network having a CLG structure, even structures that are typically amenable to inference (i.e., polytrees) [43]. For the junction tree algorithm,

any strongly rooted junction tree will contain all discrete variables within a single clique [39]. In the variable elimination setting, computing an exact posterior marginal is done by enforcing an ordering in which we marginalize away continuous (Gaussian) variables before discrete ones [15, 42]. However, if $P(\mathbf{X})$ contains two or more disjoint subgraphs, then this cost can be reduced to time exponential in the size of (number of nodes in) the largest subgraph.

In the previous work where we learned a model for multiple sensors, we handled these inference tasks in the following way. At each time step t, we computed query (3.1) as described above—by effectively enumerating all 2^N components of a piecewise Gaussian, and taking the component with the highest probability value:

$$\operatorname*{argmax}_{\vec{s}} P\left(S_1 = s_1, S_2 = s_2, \ldots, S_N = s_N | O_1 = o_1, O_2 = o_2, \ldots, O_N = o_N\right).$$

Then, we treated this MAP estimate as new evidence for the sensor states at time t and computed the updated estimate of the hidden "true" temperatures, $\mathbf{X}$,

$$P\left(X_1, X_2, \ldots, X_N | S_1 = s_1, S_2 = s_2, \ldots, S_N = s_N, O_1 = o_1, O_2 = o_2, \ldots, O_N = o_N\right).$$

Our motivation for handling inference in this two-step process was that, in an online setting, we must make a decision that each sensor at time t is working or broken rather than postponing this decision and maintaining a "belief state", i.e. 79% working and 21% broken. Not only was this approximation useful for an online QC system, it also exempted us from having to maintain an exact belief state that doubles in size after each time step. Once we determined the state of each sensor, we propagated forward $P(\mathbf{X}^t | \vec{s}^{\,t}, \vec{o}^{\,t})$ as our belief about $\mathbf{X^t}$ to time $t+1$. Thus, our approximation was made by considering only the mixture component corresponding to the MAP of the S_i variables at each time step. However, by peforming exact inference for the MAP query, our approach was limited to only relatively small deployments (10 or fewer sensors).

3.4 Methods

In this section, we describe three algorithms for approximate inference in the hybrid DBN model: Rao-Blackwellized particle filtering (RPBF), expectation propagation (EP), and

a greedy search approach to MAP inference. We focus on how each of these algorithms can be adapted to solve the inference task described in Section 3.3, and we show how to implement them for our probabilistic QC model. We begin our discussion of RBPF by providing a basic review of particle filtering and importance sampling, then describe the proposal distribution used in our QC model and the overall RBPF algorithm. Next, we provide a brief review of message-passing algorithms and then describe how expectation propagation can extend this framework to CLG models. Finally, we introduce a relatively simple (though effective) greedy search method that biases its search toward fewer sensors being *broken* at a given time step.

3.4.1 Rao-Blackwellized Particle Filtering

Rao-Blackwellized particle filtering is a stochastic, MCMC-based approach to approximate inference. The goal is to estimate some complicated posterior using a finite set of particles, where each particle is a fixed instantiation to some or all of the latent variables in the probabilistic model. For example, suppose we wanted to estimate $P(\mathbf{S}^t|\mathbf{O}^t = \vec{o})$ within a single time slice. One approach might be to draw K particles from $P(\mathbf{S}^t|\mathbf{O}^t = \vec{o})$ (a discrete distribution over 2^N possible values), $\vec{s}^{\,k,t} \sim P(\mathbf{S}^t|\mathbf{O}^t = \vec{o}), k = 1, \ldots, K$. We could then compute the posterior using an empirical estimate

$$\hat{P}(\mathbf{S}^t|\mathbf{O}^t = \vec{o}) \approx \frac{1}{K}\sum_{k=1}^{K} \delta(\vec{s}^{\,k,t}),$$

where $\delta(\vec{s}^{\,k,t})$ is the Dirac delta function located at $\vec{s}^{\,k,t}$. The expression on the right is equivalent to the average of K multinomial distributions, each of which has point mass at exactly one value: $\vec{s}^{\,k,t}$. Of course, if we actually had the posterior $P(\mathbf{S}^t|\mathbf{O}^t = \vec{o})$ from which to draw particles, we would not need an empirical estimate to approximate it. Moreover, representing the exact posterior as a distribution that can be sampled is often itself an intractable problem. Instead, RBPF relies on importance sampling [38], where a *proposal distribution* $Q(\mathbf{S}^t|\mathbf{O}^t = \vec{o})$ is substituted for the actual posterior. There are only 2 conditions for this proposal distribution: first, that it is computationally inexpensive to sample from (draw particles), and second, that it have positive mass (or density) wherever the true posterior has positive mass (or density), i.e., $P(\vec{s}^{\,k,t}|\mathbf{O}^t = \vec{o}) > 0 \Rightarrow$

$Q(\vec{s}^{\,k,t}|\mathbf{O}^t = \vec{o}) > 0$. Each particle k drawn from the proposal distribution is weighted by

$$w_k = \frac{P(\mathbf{S}^t = \vec{s}^{\,k,t}|\mathbf{O}^t = \vec{o})}{Q(\mathbf{S}^t = \vec{s}^{\,k,t}|\mathbf{O}^t = \vec{o})}.$$

If a particle's weight is large (numerator > denominator), then the particle was less likely to come from the proposal, but more likely under the true posterior; thus, we want the particle to "count" more when we estimate the posterior. If the weight is small (numerator < denominator), then we want to discount the impact of each such particle in estimating the posterior, because it is not as likely to have come from the true distribution. After the weights are normalized, denoted $\tilde{w}_k$, the empirical estimate of the posterior is

$$\hat{P}(\mathbf{S}^t|\mathbf{O}^t = \vec{o}) \approx \frac{1}{K}\sum_{k=1}^{K} \tilde{w}_k \delta(\vec{s}^{\,k,t}).$$

Rao-Blackwellized particle filtering addresses a weakness of standard particle filtering methods in that, for high-dimensional posteriors, many particles are often required to provide a reasonable estimate of the true posterior. RPBF leverages cases where the structure of a probabilistic model (set of conditional dependencies) is such that, if a subset of the latent variables were fixed (i.e., their values were sampled), we could analytically marginalize our uncertainty about the remaining latent variables, tractably [19, 50]. Thus, we need only sample from a lower dimensional space spanned by a subset of the latent variables, which ideally requires far fewer particles than the full latent space. In a CLG-setting, this typically means fixing the values of the discrete variables, because inference over strictly Gaussian variables is computationally less expensive (the cost is cubic in the number of Gaussian variables). For our case, the posterior within a single time slice $P(\mathbf{S}, \mathbf{X}|\mathbf{O} = \vec{o})$ is a 2^N-component mixture of N-dimensional multivariate Gaussians. It follows that, for even modest N, we would need a generous number of particles to accurately capture the mixing weights, means, and covariances of each mixture component. However, we note that if we fixed the values of $\mathbf{S}$, then the model reduces to a N-dimensional Kalman filter, where exact inference scales polynomially ($\Theta(N^3)$) in the number of sensors [35, 71].

In designing our proposal distribution for the QC model, we prefer that it take into account recent observations from all of the sensors at time t, so as to not generate

many particles inconsistent with the evidence (such particles would eventually be given low weight anyway). The most obvious candidate is the distribution suggested above, $P(\mathbf{S}^t|\mathbf{O}^t = \vec{o}^{\,t})$; unfortunately, this joint distribution is expensive to compute and requires 2^N probability values to represent. Instead, we approximate this posterior by removing the asynchronic edges among all process variables $\mathbf{X}$ (edges among variables X_i^t and $X_j^t : \forall i, j \in 1...N$). The intuition is that our proposal distribution is assuming that all N processes and their observations are independent of one another. Let $P'(\cdot)$ denote probability distributions associated with the proposal distribution, and let $P(\cdot)$ refer to the original distributions defined for the true model. At $t = 0$, the proposal distribution is calculated as

$$\begin{aligned} Q^0(\mathbf{S^0}|\mathbf{O}^0 = \vec{o}^{\,0}) &= \prod_{i=1}^{N} \int_{X_i^0} P'(O_i^0 = o_i^0|S_i^0, X_i^0) P'(S_i^0) P'(X_i^0) dX_i^0 \\ &= \prod_{i=1}^{N} P'(O_i^0 = o_i^0|S_i^0) P'(S_i^0) \end{aligned} \tag{3.5}$$

We begin at $t = 0$ by drawing K particles from $Q^0 : \vec{s}^{\,1,t}, \ldots, \vec{s}^{\,K,t} \sim Q^0$. Let $s_i^{k,t}$ denote the sampled value $\in \{working, broken\}$ of the sensor state variable S_i at time t associated with particle k. We draw the value of each sensor state $s_i^{k,0}$ independently from its respective $P'(S_i^0|O_i^0 = o_i^0)$ to avoid representing the full joint Q^0. Next, we compute the importance weight for each particle:

$$w_k^0 = \frac{P(O^0 = \vec{o}^{\,0}|\mathbf{S}^0 = \vec{s}^{\,k,0}) \prod_{i=1}^{N} P(S_i^0 = s_i^{k,0})}{Q^0(\mathbf{S}^0 = \vec{s}^{\,k,0}|\mathbf{O}^0 = \vec{o}^{\,0})}, \tag{3.6}$$

where $P(O^0 = \vec{o}^{\,0}|\mathbf{S}^0 = \vec{s}^{\,k,0})$ is the probability of the observation $\vec{o}^{\,0}$ under the multivariate Gaussian (MVG)

$$P(O^0|\mathbf{S}^0 = \vec{s}^{\,k,0}) = \int_{X_1^0,\ldots,X_N^0} P(\mathbf{X}^0) \prod_{i=1}^{N} P(O_i^0|S_i^0 = s_i^{k,0}, X_i^0) dX_1^0 \ldots dX_N^0.$$

For each particle drawn at time t, we store the associated sufficient statistics of the N-dimensional MVG over the process variables, $\mathbf{X}^{k,t} = P(\mathbf{X}^{k,t}|\vec{s}^{\,k,t})$. Thus, each particle contains a score, a configuration over the sensor-state variables, and the mean and

covariance matrix of the process distribution for that configuration: $(w_k^0, \vec{s}^{\,k,0}, \mu_k^0, \Sigma_k^0)$. Each particle will use its MVG at time t as the forward message to time $t+1$ in our filtering algorithm. Lastly, we normalize all of the particle weights to create a discrete distribution over our original K particles,

$$\tilde{w}_k^0 = \frac{w_k^0}{\sum_{j=1}^{K} w_j^0},$$

and then draw a new set of K particles from the resultant distribution. This resampling has the effect of discarding particles that are inconsistent with the evidence seen thus far. For example, one particle may sample a value $S_i^{k,t} = working$ for an air temperature sensor observing 50℃. $P'(S_i^t = working | O_i^t = 50)$ would be very small, but it could still be possible to draw a sample corresponding to this state with many particles. The process model for this particle will then update its belief to say that the true air temperature is 50℃. The particle carries this belief about the latent process $\mathbf{X}^{k,t}$ to $t+1$. It is likely evidence at time $t+1$ will also be inconsistent with this belief, and so the particle will be weighted lower and lower in each subsequent time step. Hence, it is best to replace these particles with more-representative particles.

The algorithm for RBPF in subsequent time steps is similar. At time t, each particle from $t-1$ draws a new value for $\mathbf{S}^t$ from the proposal distribution Q^t. The general form of our chosen proposal distribution is also similar to the one at $t=0$. We remove all asynchronic arcs among the process variables at time t; however, we maintain 1-to-1 temporal dependencies $X_i^{t-1} \rightarrow X_i^t$. Autocorrelation among each process variable does not significantly increase our burden of computation; in fact, we are now effectively maintaining N independent Kalman filters. Each particle then draws its value of the sensor-state variables at t:

$$\vec{s}^{\,k,t} \sim Q^t(\mathbf{S}^t | \mathbf{O}^{0:t} = \vec{o}^{\,0:t}, \mathbf{S}^{0:t-1} = \vec{s}^{\,k,0:t-1}),$$

where $Q^t(\mathbf{S}^t|\mathbf{O}^{0:t} = \vec{o}^{\,0:t}, \mathbf{S}^{0:t-1} = \vec{s}^{\,k,0:t-1}) =$

$$\prod_{i=1}^{N} \int_{X_i^t, X_i^{t-1}} P'(O_i^t = o_i^t|S_i^t, X_i^t) P'(S_i^t) P'(X_i^t|X_i^{t-1}) P'(X_i^{k,t-1}) dX_i^{t-1} dX_i^t$$
$$= \prod_{i=1}^{N} P'(S_i^t|O_i^t = o_i^t) P'(S_i^t). \tag{3.7}$$

The term $P'(X_i^{k,t-1})$ refers to the belief about process variable i associated with particle k at time $t-1$ under the proposal distribution, and it is akin to a forward α message in standard HMM filtering. It is a multivariate Gaussian parameterized by μ_k^{t-1} and Σ_k^{t-1}, which are functions of $\mathbf{S}^{0:t-1}$ and $\mathbf{O}^{0:t-1}$ (though this is not explicitly shown above). As before, we draw the value of each sensor state $s_i^{k,t}$ independently from its respective $P'(S_i^t|O_i^t = o_i^t)$ to avoid representing the full joint Q^t. Figure 3.4 shows the graphical models of the true and proposal distributions for an example QC model having 3 sensors. The observation model in the proposal distribution is identical to observation model in the true distribution; that is, $P'(O_i^t|X_i^t, S_i^t) = P(O_i^t|X_i^t, S_i^t)$ and $P'(S_i^t) = P(S_i^t)$. However, because our proposal only allows X_i^t to be conditioned on itself at $t-1$, $P'(X_i^t|Par(X_i^t))$ will have greater variance than $P(X_i^t|Par(X_i^t))$, suggesting that the proposal will have more diffuse belief regarding values of the latent process than the true model.

Rather than compute a new weight for the partial particle trajectories $\vec{s}^{\,k,0:t}$, we can use sequential importance sampling to update particle k's weight from $t-1$ [20]. The updated weights are computed

$$w_k^t = w_k^{t-1} \frac{P(\mathbf{O}^t = \vec{o}^{\,t}|\mathbf{O}^{0:t-1} = \vec{o}^{\,0:t-1}, \mathbf{S}^t = \vec{s}^{\,k,0:t}) \prod_{i=1}^{N} P(S_i^t = s_i^{k,t})}{Q^t(\mathbf{S}^t = \vec{s}^{\,k,t}|\mathbf{O}^{0:t} = \vec{o}^{\,0:t}, \mathbf{S}^{0:t-1} = \vec{s}^{\,k,0:t-1})}, \tag{3.8}$$

where $P(\mathbf{O}^t = \vec{o}^{\,t}|\mathbf{O}^{0:t-1} = \vec{o}^{\,0:t-1}, \mathbf{S}^t = \vec{s}^{\,k,0:t})$ is the probability of the observation $\vec{o}^{\,t}$ under the MVG

$$P(\mathbf{O}^t = \vec{o}^{\,t}|\mathbf{O}^{0:t-1} = \vec{o}^{\,0:t-1}, \mathbf{S}^t = \vec{s}^{\,k,0:t}) = \int_{\mathbf{X}^t, \mathbf{X}^{t-1}} P(\mathbf{X}^t|\mathbf{X}^{t-1}) P(\mathbf{X}^{k,t-1})$$
$$\prod_{i=1}^{N} P(O_i^t|S_i^t = s_i^{k,t}, X_i^t) d\mathbf{X}^{t-1} d\mathbf{X}^t \tag{3.9}$$

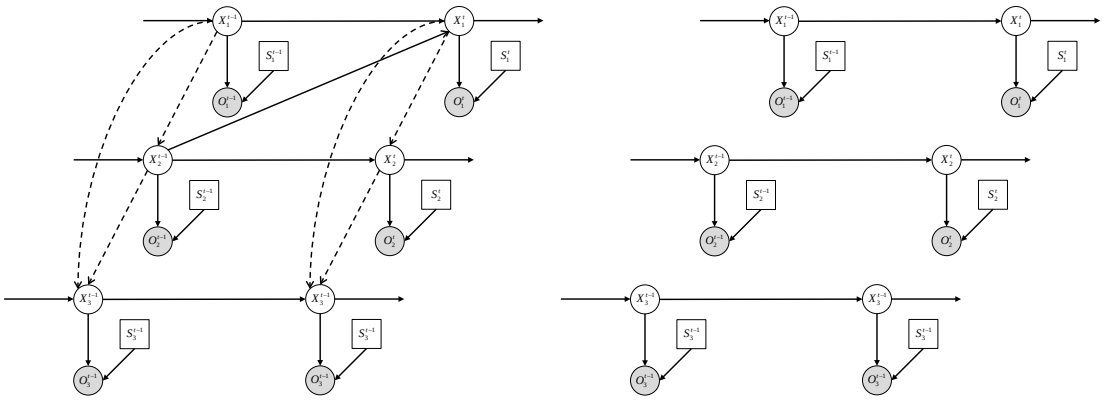

Figure 3.4: Left: The graphical model of the true QC model. Solid edges denote temporal conditional dependencies among the process variables $\mathbf{X}^{t-1}$ and $\mathbf{X}^{t}$. Dashed edges indicate asynchronic conditional dependencies among the process variables. Right: The corresponding proposal distribution for the 3-sensor QC model. Note that the only temporal edges that have been retained in the proposal are those from variables X_i^{t-1} to X_i^t, and that all asynchronic edges among the process variables have been removed.

As in equation 3.7, $P(\mathbf{X}^{k,t-1})$ refers to the belief about joint process at time $t-1$ under the true distribution, given the previous configuration of the sensor-state variables associated with particle k, $\vec{s}^{\,k,0:t-1}$. At any time step, we can obtain an approximate MAP estimate of the sensor states from time 0 to t. We do this by taking the particle k that has the highest numerator value in equation (3.8) at time t, and using its sampled values of the senor-state variables, $\vec{s}^{\,k,0:t}$. Similarly, we can obtain an imputation of the true observation in cases where we believe a sensor is broken by examining the mean and variance of $\mathbf{X}_t^k$. Somewhat less intuitive is that the MAP particle may change from time t to time $t+1$, as one particle becomes more likely with the new evidence. The new MAP particle k' may have different values of the sensor-state variables in times $0:t$, in effect changing our belief about already-diagnosed sensor observations.

3.4.2 Expectation Propagation

Like belief propagation (BP), expectation propagation (EP) is a deterministic, message-passing algorithm for approximate inference in probabilistic models. Message-passing algorithms perform inference using localized messages sent among the variables in a

factor-graph representation of the probabilistic model [54]. There are two types of messages used in this scheme: messages from a variable to its adjacent factors $\phi_{X_j \to f_j}$, and messages from factors to adjacent variables $\phi_{f_j \to X_i}$. Here, we define a factor and variable to be *adjacent* if they share a common edge in the factor graph.

In a single iteration of the algorithm, each variable sends a message to its adjacent factors that reflects the current belief about that variable,

$$\phi_{X_i \to f_j} = \prod_{f_n \in fac(X_i), f_n \neq f_j} \phi_{f_n \to X_i}, \tag{3.10}$$

where $fac(X_i)$ is the set of all factors that are adjacent to the variable X_i. Each factor also sends a message to its adjacent variables,

$$\phi_{f_j \to X_i} = \sum_{X_n \in dom(f_j), X_n \neq X_i} f_j(X_n) \prod_{X_m \in dom(f_j), X_m \neq X_i} \phi_{X_m \to f_j}, \tag{3.11}$$

where $dom(f_j)$ is the set of variables spanned by factor f_j (all variables adjacent to f_j in the factor graph). The message from factor f_j to variable X_i is calculated by taking the product of all incoming messages from the neighbors of f_j, and then marginalizing away all variables $X_n \in dom(f_j)$ except for X_i. The expression $f_j(X_n)$ is used instead of $P(X_n)$ because, strictly speaking, the factor may represent a potential over the variable X_n rather than a proper probability distribution. In effect, the factor f_j collects all incoming messages regarding the belief about its adjacent variables, and then prepares a message for each of those variables $X_i \in dom(f_j)$ by reducing the scope of that message to only X_i through marginalization.

A message-passaging schedule determines the ordering in which factors and variables calculate/propagate their messages, with each message being updated in a single iteration of message passing. The BP algorithm ends when both sets of messages converge or after a fixed number of iterations. For tree-structure graphs, BP is known to converge to the exact solution; however, for more general graphical structures, BP may converges to an approximation of the true posterior, or it may not converge at all. The posterior marginal $P'(X_i)$ at the variable X_i is then calculated by taking the product of all incoming messages to that variable

$$P'(X_i) = \prod_{f_n \in fac(X_i)} \phi_{f_n \to X_i}.$$

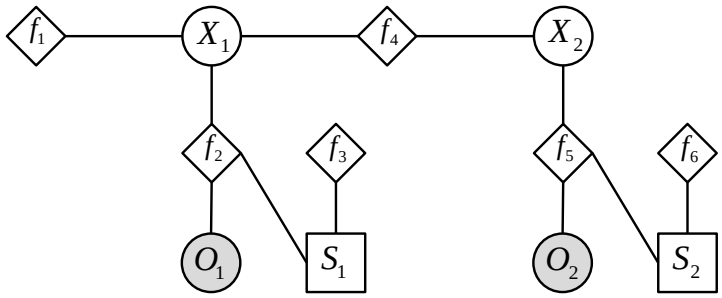

Figure 3.5: A factor-graph representation of a two-sensor model for 1 time slice. Factor nodes are denoted by diamonds, continuous variables by circles, and discrete variables by squares.

Figure 3.5 shows a factor graph for a two-sensor QC model for one time slice. We note that there is a separate factor for each conditional distribution $P(Y|X)$in the original directed graphical model, and a separate factor for each prior distribution $P(X)$. Consider the message $\phi_{f_2 \to X_1}$:

$$\begin{aligned}
\phi_{f_2 \to X_1} &= \sum_{S_1} f_2(O_1 = o_1 | X_1, S_1) \prod_{X_m \in \{O_1, S_1\}} \phi_{X_m \to f_2} \\
&= \sum_{S_1} f_2(O_1 = o_1 | X_1, S_1) \phi_{O_1 \to f_2} \phi_{S_1 \to f_2} \\
&= \sum_{S_1} f_2(O_1 = o_1 | X_1, S_1) P(S_1),
\end{aligned}$$

where $\phi_{S_1 \to f_2} = P(S_1)$ and $\phi_{O_1 \to f_2} = 1.0$.

There are two issues worth noting regarding $\phi_{f_2 \to X_1}$. First, it is not a proper distribution, and thus has no moment form (explicit mean and variance). This is because it is a conditional linear-Gaussian conditioned on X_1, and $P(X_1)$ is not factored in until the message $\phi_{X_1 \to f_4}$ is computed. Alternatively, we could make this a proper distribution by enforcing a messsage schedule where X_1 first passes a message to f_2. The second issues is that this message will not become any less compact after marginalizing away S_1. In order to maintain exactness of inference, we must keep the parameters of the conditional distribution $P(O_1|X_1, S_1)$ for both $S_1 = working$ and $S_1 = broken$. Here, this means maintaining the means, variances, and weights of $P(O_1|X_1, S_1)$ and the probability values of $P(S_1 = working)$ and $P(S_1 = broken)$. After $\phi_{X_1 \to f_4}$ is computed, we can marginalize

away S_1; however, the result will still be a mixture of Gaussians (a proper density, but having no fewer parameters). In general, messages among the process variables and their corresponding factors have size exponential in the number of sensors N.

Expectation propagation, developed by Minka [47], offers a solution to the second issue by inserting an intermediate projection step after marginalization occurs in the messages computed in (3.11). The projection step comes from assumed density filtering (ADF) [45], wherein the goal is to identify the best approximation within a specific family of distributions (here, the exponential family) to the true posterior distribution (a mixture of Gaussians). Best, in this case, means optimizing to achieve the minimal KL-divergence between the approximated posterior and true posterior. For approximating a mixture of Gaussians with a single Gaussian, Lauritzen shows that the best approximation can be computed using the collapsed Gaussian described in Equations (3.3, 3.4).

Our application of EP to the inference task in (3.2) is motivated by its similarity to Minka's "clutter problem" [48]. In the clutter problem, the goal is to estimate the mean parameter μ of a target Gaussian distribution using observations from that Gaussian intermixed with unrelated clutter. In our case, the target Gaussian is the process model, and we have observations from this Gaussian in the form of readings from *working* sensors that are similarly interspersed with clutter generated by *broken* sensors. Before we introduce our application of EP, we note that we can generalize the scope of individual variable nodes to clique nodes in a factor graph, where a clique represents a set of variables. In the factor graph representation of our QC model (shown in Figure 3.6), we use $N+1$ such cliques at each time slice, where clique $\psi_0 = \mathbf{X}^t$ and cliques $\psi_i = \left\{S_i^t, O_i^t\right\}$ for $i = 1, \ldots, N$. The factors $f_i, i = 1, \ldots, N$ correspond to the conditional distributions $P(O_i^t|S_i^t, X_i^t)P(S_i^t) = P(S_i^t, O_i^t|X_i^t)$ and f_0 corresponds to the prior distribution on the process $P(\mathbf{X}^0)$ if $t = 0$; otherwise, $f_0 = P(\mathbf{X}^t|\mathbf{X}^{t-1})$.

In Minka's work, the observations are distributed $x_1, \ldots, x_n \sim (1-w)N(x; \mu, \sigma_a^2) + wN(x; 0, \sigma_b^2)$, where σ_a^2 and σ_b^2 are known. We can place a Gaussian prior $\mathrm{P}(\theta) \sim N(0, \sigma^2)$ on μ, and then model the joint distribution $P(\mathbf{x}, \theta) = P(\theta)\prod_{i=1}^{N} P(x_i|\theta)$ using an assumed density filter. The ADF resembles a Kalman filter where we begin with our prior $P(\theta)$, and then update it using a single observation x_1. The resultant true posterior $P'(\theta)$ is a Gaussian mixture, which we then project onto an approximating distribution (a single Gaussian). The process then repeats for $x_2, \ldots, x_n$. Of course, the final posterior on θ depends on the order in which we processed the observations. The EP algorithm attempts

to remove this dependency on the observation ordering. In EP, the approximation occurs in the update x_i sends to $P(\theta)$ so that $P'(\theta)$ can be computed exactly, rather than treating the update exactly and approximating the resultant posterior. This difference may appear subtle, but it facilitates the iterative updating of the posterior used in message-passing algorithms.

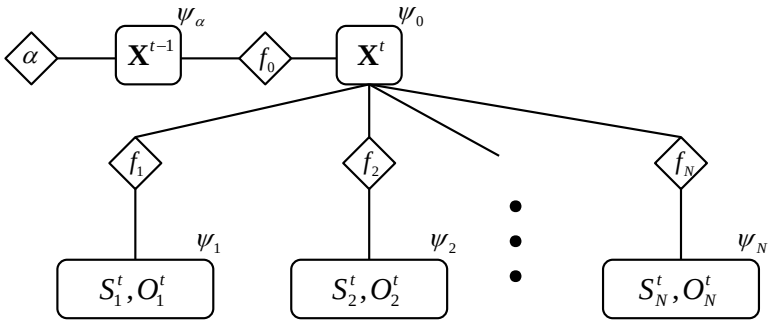

Figure 3.6: A factor-graph representation of the QC model for 1 time slice used in our EP algorithm. Factor nodes are denoted by diamonds. Rounded-edge rectangles denote cliques of variables.

The application of EP to the QC model proceeds as follow. We first compute the apriori belief about the process model $P(\mathbf{X}^t)$ by sending a message from f_0 to $\mathbf{X}^t$

$$\begin{aligned}
\phi_{f_0 \to \psi_0} &= \int_{\mathbf{X}^{t-1}} f_0(\mathbf{X}^t|\mathbf{X}^{t-1}) \phi_{\psi_\alpha \to f_0} d\mathbf{X}^{t-1} \\
&= \int_{\mathbf{X}^{t-1}} f_0(\mathbf{X}^t|\mathbf{X}^{t-1}) \phi_{f_\alpha \to \psi_\alpha} d\mathbf{X}^{t-1} \\
&= \int_{\mathbf{X}^{t-1}} f_0(\mathbf{X}^t|\mathbf{X}^{t-1}) f_\alpha(\mathbf{X}^{t-1}) d\mathbf{X}^{t-1} \\
&= P(\mathbf{X}^t).
\end{aligned}$$

This message is our belief about the current process before we have seen any observations at time t; that is, we have only propagated $\mathbf{X}^{t-1}$ through the transition model $P(\mathbf{X}^t|\mathbf{X}^{t-1})$ captured in the factor f_0. The factor f_α represents the prior distribution $P(\mathbf{X}^{t-1})$, and we discuss how to calculate it in Section 3.4.2.1. Under our message-passing schedule, the next message is sent from ψ_0 to one of the factors f_i $(i \in 1, \ldots, N)$, where it is

combined with the observation o_i^t, and then returned in order to update the joint process distribution $P(\mathbf{X}^t)$.

$$\begin{aligned}
\phi_{f_i \to \psi_i} &= \int_{\mathbf{X}^t \setminus X_i^t} f_i(S_i^t, O_i^t | X_i^t) \phi_{\psi_0 \to f_i} d\mathbf{X}^t \setminus X_i^t \\
&= \int_{\mathbf{X}^t \setminus X_i^t} f_i(S_i^t, O_i^t | X_i^t) P(\mathbf{X}^t) d\mathbf{X}^t \setminus X_i^t \\
&= P(S_i^t, O_i^t | X_i^t) P(X_i^t) \\
&= P(S_i^t, O_i^t, X_i^t). \qquad (3.12)
\end{aligned}$$

The form of this message is a table of multivariate Gaussians, where each Gaussian is 2D (having a domain of X_i^t and O_i^t), and there is a separate table entry for each value for S_i^t. A message is then sent back from ψ_i that incorporates the observation $O_i^t = o_i^t$ at that clique. By observing O_i^t, we now have a table containing two univariate Gaussian distributions over X_i^t. The ADF approximation is then applied to the return message

$$\begin{aligned}
\phi_{f_i \to \psi_0} &= \frac{ADF(\sum_{S_i^t} P(S_i^t, O_i^t = o_i^t, X_i^t))}{\phi_{\psi_0 \to f_i}} \\
&= P'(X_i^t),
\end{aligned}$$

which results in a single Gaussian distribution for $P'(X_i^t)$ rather than the true posterior—a mixture of two Gaussians. Figure 3.7 shows a visualization of this message-passing procedure for a single sensor clique. Note that the Gaussian for $S_i^t = \textit{broken}$ in step 2.) is identical to the initial $P(X_i^t)$ sent from the process clique in step 1.). This is because, when the sensor is *broken*, observations are disconnected from the process model, so that they provide no information to update the belief about X_i^t. The posterior $P'(X_i^t)$ is combined with the process model $P'(X_1^t, \ldots, X_N^t) = P(X_1^t, \ldots, X_N^t) P'(X_i^t)$, resulting in an updated belief about $X_1^t, \ldots, X_{i-1}^t, X_{i+1}^t, \ldots, X_N^t$ due to the covariance structure captured in $P(\mathbf{X}^t)$. A new message is then generated and sent to clique ψ_2, and the process is repeated until the last message $\phi_{f_N \to \psi_0}$ is absorbed by the process model.

One iteration of EP is equivalent to performing asynchronic inference using an assumed density filter. That is, our final distribution on $P(\mathbf{X}^t)$ and the sensor states $P(\mathbf{S}^t | \mathbf{O}^t)$ will be influenced by the order in which messages were sent/received from the

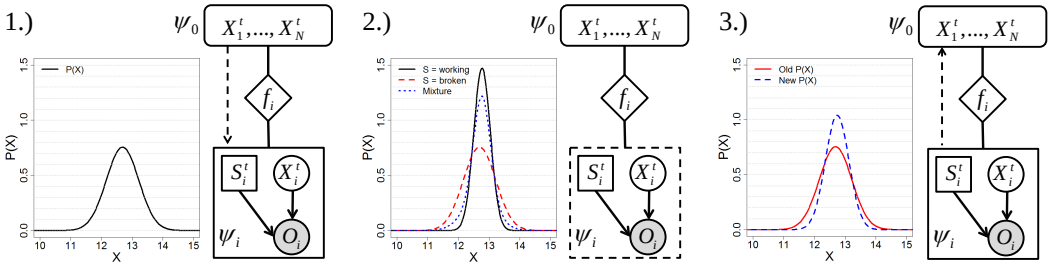

Figure 3.7: 1.) The process clique sends a message to ψ_i regarding its belief about the process variable X_i^t. The sensor clique ψ_i updates this belief with the observation $O_i^t = 12.8$ ℃. 2.) The joint distribution $P(X_i^t, S_i^t, O_i^t = 12.8)$ is represented by 2 separate Gaussians, one for the case where $S_i^t = working$ (solid black line) and one for $S_i^t = broken$ (dashed red line). The weights for the *working* and *broken* Guassian mixtures are .65 and .35, respectively. After marginalization, the true posterior is a mixture of Gaussians (dotted blue line). In 3.), this mixture is approximated by a collapsed distribution (dashed blue line).

sensors; i.e., the order in which we processed our observations. To reduce this effect, EP uses multiple iterations. Subsequent iterations are similar to the first; however, before a clique/factor generates a new message, it must "forget" the message it received from the intended recipient in the previous iteration. For example, in the second iteration, the message $\phi_{f_1 \to \psi_0}$ is calculated

$$\begin{aligned}
\phi_{f_i \to \psi_i} &= \int_{\mathbf{X}^t \setminus X_i^t} f_i(S_i^t, O_i^t | X_i^t) \frac{\phi_{\psi_0 \to f_i}}{\phi_{f_i \to \psi_0}} d\mathbf{X}^t \setminus X_i^t \\
&= \int_{\mathbf{X}^t \setminus X_i^t} P(S_i^t, O_i^t | X_i^t) \frac{P(\mathbf{X^t})}{P'(X_i^t)} d\mathbf{X}^t \setminus X_i^t \\
&= P(S_i^t, O_i^t | X_i^t) P(X_i^t) \\
&= P(S_i^t, O_i^t, X_i^t),
\end{aligned}$$

where $P'(X_1^t)$ is the message $\phi_{f_1 \to \psi_0}$ in the previous iteration of message passing. To provide some intuition about this new message, consider that

$$\frac{P(\mathbf{X}^t)}{P'(X_i^t)} = \prod_{f_j \in fac(\psi_0), f_j \neq f_i} \phi_{f_j \to \psi_0}.$$

In other words, we are creating an "old posterior" over the process variables by forgetting the update given by the observation $O_i^t = o_i^t$ in the first iteration; however, this posterior still reflects observations $O_1^t = o_1^t, \ldots, O_{i-1}^t = o_{i-1}^t, O_{i+1}^t = o_{i+1}^t, \ldots, O_N^t = o_N^t$. Moreover, our belief in the previous iteration regarding whether each observation $o_{j\neq i}^t$ came from a *working* or *broken* sensor was influenced by $P'(X_i^t)$, and those beliefs remain unchanged until they are revisited in the second iteration of message-passing. For this reason, the "forgetting" step is not complete; we retain the context $P'(X_i^t)$ provided for the other other $N-1$ observations.

To get a new belief state at the sensor clique ψ_i, we divide out the message received from the process clique ψ_0 in the previous iteration:

$$P(\psi_i) = \frac{P(S_i^t, O_i^t = o_i^t | X_i^t) P(X_i^t)}{P'(X_i^t)}.$$

The message from the sensor clique ψ_i to the process clique ψ_0 is then calculated as before.

$$\begin{aligned} \phi_{f_i \to \psi_0} &= \frac{ADF(\sum_{S_i^t} P(S_i^t, O_i^t = o_i^t, X_i^t))}{\phi_{\psi_0 \to f_i}} \\ &= P'(X_i^t). \end{aligned} \tag{3.13}$$

A known artifact of the EP algorithm arises in our application of it to the QC problem; namely, division of a Gaussian density by another Gaussian density may result in a Gaussian having a negative variance. In our case, this occurs when we create new messages from the sensor cliques to the process clique (3.13). To understand why this occurs, consider first that division among two exponential family distributions of the same type is performed by taking the difference of their canonical (exponential) forms. For Gaussians, the canonical form of the covariance matrix is the precision matrix Σ^{-1}. Hence, the resultant variance of the Gaussian $P(C) = \frac{P(A)}{P(B)}$ is $\Sigma_C^{-1} = \Sigma_A^{-1} - \Sigma_B^{-1}$, and so $\Sigma_C < 0$ whenever B has smaller variance than A. Translating to our case, if an observation makes the model less certain about the value of the latent process than the prior belief on the process

$$Var\left(ADF\left(\sum_{S_i^t} P(S_i^t, O_i^t = o_i^t, X_i^t)\right)\right) > Var(\phi_{\psi_0 \to f_i}),$$

then we obtain a negative variance; however, it does not necessarily mean the algorithm is behaving incorrectly. When we multiply this negative-variance Gaussian onto the process model, we increase the process model's uncertainty (increase its variance) regarding X_i^t. Multiplication of two exponential family distributions can be handled by addition of their canonical forms: $P(C) = P(A)P(B) \Rightarrow \Sigma_C^{-1} = \Sigma_A^{-1} + \Sigma_B^{-1}$. If $\Sigma_B^{-1} < 0$, then Σ_A^{-1} becomes smaller, and consequently, Σ_A gets larger. More intuitively, observations about which the model is uncertain increase the variance in the estimate of the true process. Still, this can cause numerical instability issues later when we try to compute an old posterior by forgetting the last message sent from f_i to ψ_0. To avoid these issues, we set the mean and variance of $\phi_{f_i \to \psi_0}$ to 0.0 and 10^6 whenever the variance of this message is < 0 (i.e., the message can be uninformative, but it cannot increase the variance of $P(X_i^t)$).

We consider three different message-passing schedules in our implementation of EP. Before initiating message-passing in a new time slice t, we first sort the sensor cliques $\psi_1, \ldots, \psi_N$ according to their initial belief about the *working*/*broken* status of their sensor-state variables. This is done by propagating the apriori state of the process $P(X_t|\vec{o}^{\,0:t-1})$ to each of the sensor cliques through the message $\phi_{f_i \to \psi_i}$ in Equation (3.12). Note that we are only sending the process to each sensor; we are not yet updating the process based on the observations o_i^t at each of the cliques. We then calculate the "confidence" c_i at each clique

$$c_i = \left| P(S_i^t = working | O_i^t = o_i^t, X_i) - P(S_i^t = broken | O_i^t = o_i^t, X_i) \right|,$$

and sort the cliques based on their confidence scores. The sorted ordering is used for the remainder of message-passing at time t; the confidences are not re-computed between iterations of message passing. We consider three message-passing schedules: *low-to-high*, where messages are exchanged between cliques in increasing order of confidence (messages are first exchanged with cliques uncertain about the state of the sensor); *high-to-low*, where messages are exchanged in decreasing order of confidence; and no preference, where the message passing schedule is given by the ordering of sensors in the QC model $\psi_1, \psi_2, \ldots, \psi_N$.

3.4.2.1 MAP Inference using EP

Thus far, we have not discussed how we compute the MAP sensor state configuration using EP within a time slice. Here, we describe a few approaches to this computation. Later, we provide an analysis of these approaches in Section 3.6.

- *Marginal MAP*: We perform EP as described above until convergence in the message parameters or up to a maximum of five iterations. The final value of each sensor state variable S_i^t is then computed as

$$\underset{s_i^t}{\operatorname{argmax}} P(S_i^t, O_1^t = o_i^t, X_i^t)$$

  for the sensor clique ψ_i.

- *Iterative MAP*: We perform EP as described above until convergence in the message parameters or up to a maximum of five iterations. Then, based on the message-passing schedule, we fix the value of the first sensor-state variable in the schedule S_i^t according to its MAP marginal value and propagate a new message $\phi_{f_i \to \psi_0}$ to the process clique. This message reflects the fixed value of the sensor-state variable in clique ψ_i, and thus requires no collapsing approximation. The process distribution is updated with this message, and we repeat this for the second sensor clique in the message-passing schedule ψ_j. We continue in this fashion until all sensor-state variables are fixed.

- *Max-Product MAP*: This variant does not actually use EP, but rather estimates the MAP using the max-product form of belief propagation. This is achieved by replacing the summation term in equation 3.11 with $\operatorname{argmax}_{s_i^t}$. We do not need to perform the ADF approximation, because now we are only using one component of the Gaussian mixture (the one associated with the most likely sensor state), and so can pass messages exactly.

The forward alpha message over the process variables $P(\mathbf{X}^t)$ sent to time $t+1$ is dependent on how the MAP sensor configuration is calculated. For both *Marginal MAP* and *Iterative MAP*, the calibrated belief $P(\mathbf{X}^t)$ is sent to time $t+1$ after EP has converged (or reached max iterations); none of the sensor-state variables are fixed when this

distribution is computed. For *Max-Product MAP*, we send the process variable distribution corresponding to the MAP configuration of sensors, which is determined after message-passing has terminated.

3.4.3 SearchMAP

In anomaly-detection domains, anomalies are often assumed to be much less common than normal observations. This carries over into QC for ecological sensors, where we typically assume the sensors are *working* more often than they are *broken*. Our third approach to approximate inference builds on this assumption, and uses a greedy hillclimbing search to explore scenarios of increasing numbers of sensor failures. The algorithm, *SearchMAP*, begins by assuming that observations at time t are associated with every sensor $S_i^t = working$. It then determines if the observation $\vec{o}^{\,t}$ would be any more likely if just one of the sensor-state variables was flipped, $S_j^t = broken$. This is accomplished by searching over all such configurations of the sensor states where one sensor is *broken*. If so, the search procedure repeats, this time checking if $\vec{o}^{\,t}$ would be any more likely if S_j^t and one additional sensor were set to *broken*, and so on. Like EP, *SearchMAP* is a deterministic algorithm for approximate probabilistic inference.

At each time step, the algorithm begins by first computing the likelihood of the most recent observation $\mathbf{O}^t = \vec{o}^{\,t}$ under the all-working configuration of sensors,

$$maxScore = P(o_1^t, \ldots, o_N^t | S_1^t = working, \ldots, S_N^t = working) \prod_{i=1}^{N} P(S_i = working).$$

The first term is equivalent to the probability of the observations under the MVG computed by fixing all of the sensor-state variables and integrating away the process variables $\mathbf{X}_t$. This MVG is calculated as

$$\int_{\mathbf{X}^t} P(X_1^t, \ldots, X_N^t) \prod_{i=1}^{N} P(O_i^t | S_i^t = working, X_i^t) d\mathbf{X}^t.$$

We save the score of this initial configuration as *maxScore* and the configuration itself as $maxConfig = \{S_1^t = working, \ldots, S_N^t = working\}$. Next, we consider all configurations where just one of the sensor-state variables is *broken*, and compare each of their

likelihoods to *maxScore*. Each configuration's likelihood is calculated as

$$score_i = P(o_1^t, \ldots, o_N^t | S_i^t = broken, S_{j \neq i}^t = working) P(S_j = broken) \prod_{j \in 1,\ldots,N, j \neq i} P(S_j = working),$$

where $P(o_1^t, \ldots, o_N^t | S_i^t = broken, S_{j \neq i}^t = working) =$

$$\int_{\mathbf{X}^t} P(X_1^t, \ldots, X_N^t) P(O_i^t | S_i^t = broken, X_i^t) \prod_{j \in 1,\ldots,N, j \neq i}^{N} P(O_i^t | S_i^t = working, X_i^t) d\mathbf{X}^t.$$

If none of the new configurations yields an improvement to $maxScore$, then the algorithm terminates and returns the original configuration (all sensors *working*) and score. Otherwise, we choose the configuration j that increases the likelihood the most, set $maxScore = score_j$, $maxConfig$ to the associated sensor configuration, and repeat the search. In the second iteration of the search, the algorithm considers scenarios where at most 2 of any of the sensor-state variables are *broken*. After 3 or more iterations of the search, the algorithm may consider "backward steps", where it flips the *broken* state of sensor assigned in previous iterations of the search back to *working*; however, it cannot undo the flip made in the immediate previous step of the algorithm, as we already know that this would yield a lower score than the current configuration. Figure 3.8 shows a visualization of the algorithm applied to a 3-sensor QC model. The value of the process model passed forward to time $t + 1$ is given as $P(\mathbf{X}^t | \mathbf{S}^t = maxConfig, \mathbf{O}^t = \vec{o}^{\,t})$.

At each iteration, the algorithm must evaluate $O(N)$ potential configurations. Each evaluation requires time roughly cubic in the number of sensors. Because the algorithm is allowed to take backsteps (undoing previously *broken* sensors), the search may potentially explore every configuration in the worst case. Hence, the algorithm has worst-case runtime $O(2^N N^3)$. However, our empirical results indicate that the algorithm often reaches a local maxima and terminates using much fewer evaluations. Further, one could remove the backstep operator at the cost of accuracy, in which case the algorithm will only explore order $O(N^2)$ possible states.

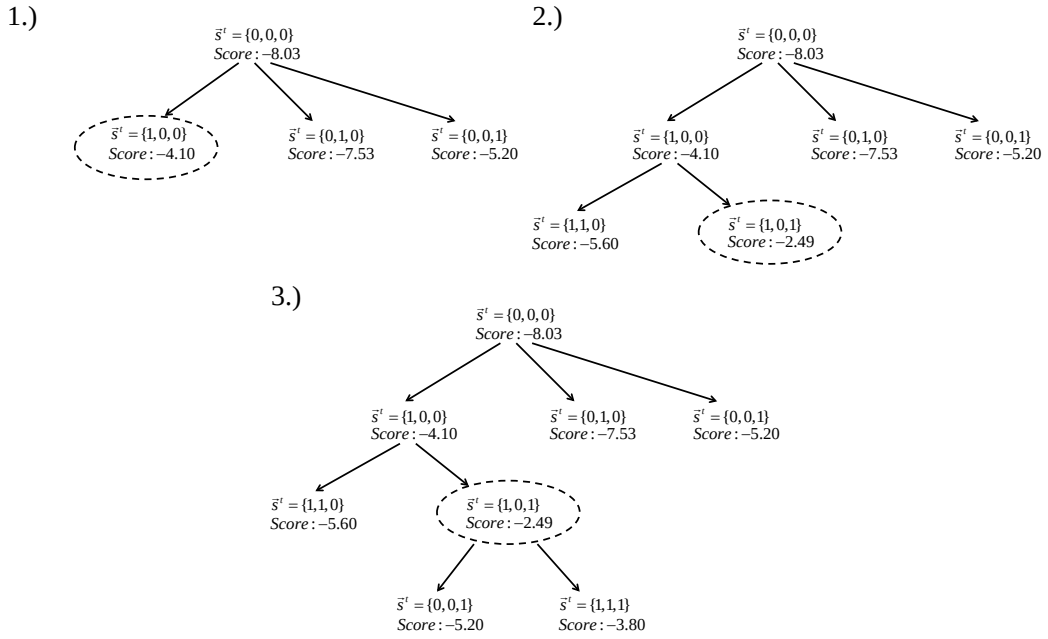

Figure 3.8: 1.) The search begins by exploring all hypotheses where one sensor is *broken*. We use bit strings to represent the status of the sensor-state variables, where 0 in the i^{th} bit denotes $S_i^t = working$ and 1 indicates $S_i^t = broken$. Breaking sensor S_1^t results in the best likelihood among all current hypotheses. 2.) The algorithm now explores breaking one additional sensor: sensors S_2^t or S_3^t. The likelihood is further improved by setting sensor S_3^t's state to *broken*. 3.) In the final iteration, the model explores undoing the step it made at iteration 1.), and breaking the last remaining *working* sensor. The first change yields a configuration the model has already explored, and setting S_2^t to broken yields no further improvement. Thus, the best configuration remains $\vec{s}^{\,t} = \{broken, working, broken\}$.

3.5 Data

Our work is motivated by the need to automate data QC for environmental sensor networks. To that end, we evaluate the inference algorithms in Section 3.4 using data collected from a sensor network located in the H.J. Andrews Experimental Forest[1], a Long Term Ecological Research (LTER) site located in the western Cascade Range of Oregon. The site is instrumented with hundreds of heterogeneous sensors distributed in a variety

[1] `http://andrewsforest.oregonstate.edu/`

of micro-climates. We focus on data collected from four meteorological benchmark sensor towers known as Primary, Central, Upper Lookout, and Vanilla Leaf. Figure 3.9 displays the Andrews LTER site and the 4 benchmark stations used in our experiments.

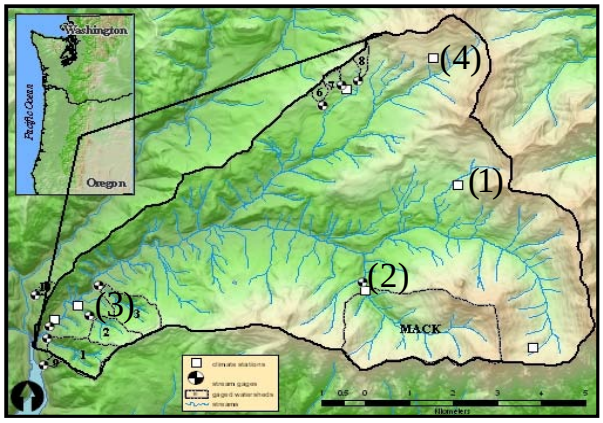

Figure 3.9: The H.J. Andrews Experimental Forest: (1) Central Met. station (elevation: 1005m), (2) Upper Lookout Met. station (elevation: 1280m), (3) Primary Met. station (elevation: 430m), (4) Vanilla Leaf Met. station (elevation: 1273m).

We focus on a set of 27 sensors distributed among these meteorological stations. For each tower, we examine 4 air temperature sensors located at approximately $1.5, 2.5, 3.5$ and 4.5m of height above the ground. Additionally, we examine a single solar radiation sensor and anemometer (mean wind speed) sensor on each of the towers. Lastly, we include one precipitation gauge from the Central met. station and two precipitation gauges from the Upper Lookout tower. These towers and sensors were specifically chosen because they have been in place from 1996 to present, have collected data at regular sampling frequencies, and the site manager was able to provide us annotations regarding maintenance and calibration of the sensors. In addition to having access to the raw observations from each of these sensors, we also have labels provided by a domain expert indicating the quality of the sensor's observation ("clean" observations coming from *working* sensors, and "suspect" values coming from *broken* sensors).

For the air temperature and solar radiation sensors, the measurements are recorded every 15 minutes. For the precipitation data, samples are collected at 5 minute intervals; however, we sum every 3 consecutive, non-overlapping readings to create 15-minute

cumulative measurements using the GLITCH system [10]. For the wind-speed data, samples represent hour-long averages. We linearly interpolate the hour-long readings to "upsample" the data to the 15-minute sampling frequency. The linear interpolation is handled in the usual way, where each quarter-hour measurement is calculated

$$y_{h+qh} = \frac{y_{h+1} - y_h}{4} * (qh) + y_h.$$

y_{h+qh} denotes the mean wind at hour h and quarter-hour qh, and y_{h+1} denotes the average wind measurement in the next hour interval. Figure 3.10 is a time series plot of readings collected from 7 sensors on the Central met. tower over a 30-day period.

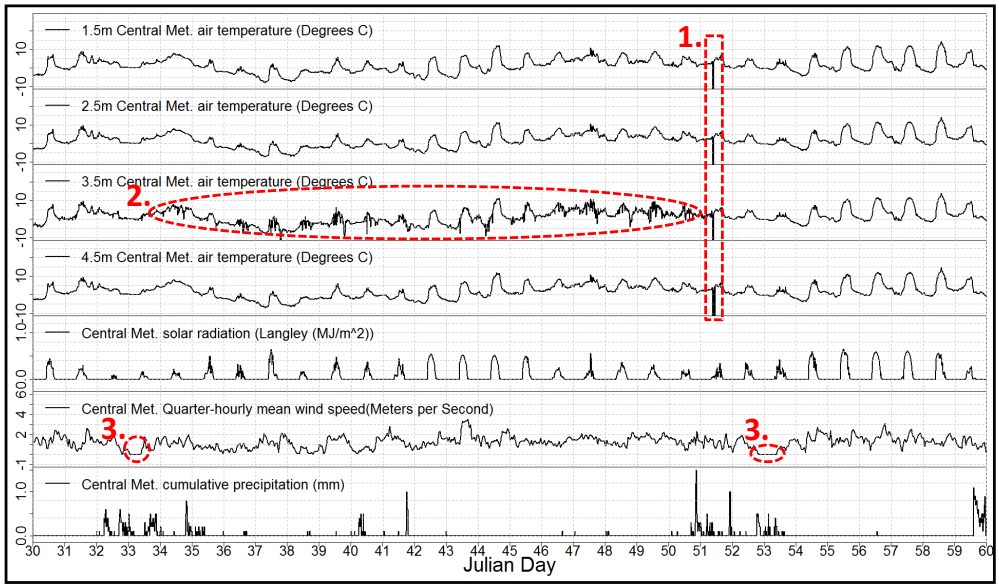

Figure 3.10: Observations from 7 sensors located on the Central benchmark meteorological tower, including 4 air temperature thermometers, a solar radiation sensor, anemometer, and precipitation gauge. This plot contains a few examples of data-anomalies caused by real sensor failures. (1.) A logger attached to the temperature thermometers malfunctions, causing it to record -53.5 ℃ for all 4 air temperature sensors; (2.) The 3.5m air temperature thermometer records erratic spikes in the temperature; (3.) The anemometer located on the top of the tower freezes due to cold temperatures, and consequently, observers zero mean wind speed.

3.5.1 Synthetic Dataset

The domain expert may have difficulty distinguishing among observations at the 15-minute frequency, and so will label "buffer" segments before and after a period where a sensor is considered *broken* to insure no suspect data is released to the public. As a result, the ground-truth labeling is biased toward over-labeling "suspect" observations. Because our model operates at the 15-minute scale, a direct comparison to the ground truth will penalize our model for failing to diagnose observations in these buffers as data-anomalies. To avoid this penalization, we create a synthetic data set that contains injected data-anomalies so that we have an exact ground truth at the same temporal scale as our QC model. We analyze our performance on both synthetic and real data sets.

To construct the synthetic data, we first find 10 non-overlapping folds of clean-only data (observations containing no "suspect" values among all 27 sensors in a single time slice) from the original dataset. We searched the dataset from 1997–2008 for the 10 longest contiguous blocks of clean data $F_1, F_2, \ldots, F_{10}$. Contiguous data are required to train the Markov component of our process model. The length of the segments (number of quarter hours) was set to the length of the smallest contiguous block: 2000 observations (approximately 21 consecutive days). For those blocks that contained more than 2000 observations, we selected a sub-block of contiguous 2000 observations, with a preference toward spanning parts of the calendar year not covered by the other segments. Next, for each fold of clean data F_i, we constructed a noise-injected fold F_i' that contains potentially three different injected anomaly types:

- *Type 1, Spike anomaly*: Gaussian noise $N(0, \sigma^2_{O_j})$ is added to the observation o_j^t, where $\sigma^2_{O_j}$ is the sample variance of the clean observations from sensor i in the current fold F_i.

- *Type 2, Bias anomaly*: The current observation o_j^t is multiplied by 1.35 to represent a biased sensor observation (for example, a broken sun shield on a thermometer might cause air temperature readings to be 35% higher than normal).

- *Type 3, Flatline anomaly*: The current observation o_j^t is set to the sensor's observation in the previous time step, o_j^{t-1}.

- *None*: The original clean observation o_j^t is used.

These synthetic anomaly types are not meant to represent all possible failure cases for sensors at the H.J. Andrews, but rather a subset of the errors described to us by the site managers. The above noise was injected following a geometric duration distribution applied independently to each sensor. We imagine this distribution behaving as a first-order Markov process with parameterization shown in Figure 3.1. For example, if the observations o_i^t was injected with a *Type 2* (bias) fault, then there is a 95% chance that o_i^{t+1} will have a *Type 2* fault, and a 5% probability it will return to normal.

	$\mathbf{P(Type^{t+1}|Type^t)}$			
Type	$None^{t+1}$	$Spike^{t+1}$	$Bias^{t+1}$	$Flatline^{t+1}$
$None^t$	.970	.010	.010	.010
$Spike^t$	.200	.700	.050	.050
$Bias^t$	.050	.025	.950	.025
$Flatline^t$	.050	.025	.025	.950

Table 3.1: The table displays the probability of a given anomaly-type being injected given the anomaly type in the previous time step.

The final product is a dataset consisting of 10 folds of clean data, $F_1, \ldots, F_{10}$, and their noise-injected equivalent folds $F'_1, \ldots, F'_{10}$.

3.6 Experiments

In this section, we evaluate the performance of the three approximate inference algorithms (as well as some minor variants on these algorithms) described in Section 3.4 in our probabilistic QC model. One of our goals is to measure the benefits (in terms of accuracy in the QC task) gained through the inclusion of additional sensors into the model. In Section 3.6.2, we assess this gain by first constructing a series of increasingly large networks, and then applying each of the three algorithms to these networks. Next, we examine the results of the three algorithms on a full network of 27 sensors. This analysis is performed on both synthetic and real data sets. We discuss some results of our analysis and provide some insight into the behavior of each algorithm.

3.6.1 Learning Process Models

Many of our experiments involve, as a preliminary step, learning a linear-Gaussian network structure to represent the process distribution $P(\mathbf{X}^t, \mathbf{X}^{t-1})$. In Section 3.3.1, we provide a brief description of how this learning occurs, but refer the reader to previous work [18] for a fuller explanation. Here, we provide specific details on the parameters used to learn each process model, and list some minor variations on the previous work that were made for this article.

Previously, structure learning was applied to learn the asynchronic (spatial) component of the process model $P(\mathbf{X}^t)$; that is, each variable X_i^t could only consider other variables at time t as candidate parent variables, such that $Par(X_i^t) = \left\{ X_j^t | X_j^t \in \mathbf{X}^t \setminus X_i^t \right\}$. The learned asynchronic process model belongs to a Markov equivalence class (MEC), which is a set of directed graphical models that all represent the same set of conditional dependencies. Thus, to determine the complete process model $P(\mathbf{X}^t, \mathbf{X}^{t-1})$, we iterated over all members of that class. In each iteration, we added X_i^{t-1} to each respective variable's set of parents (i.e., conditioned each X_i^t on itself in the previous time step) and scored the likelihood of a hold-out dataset on the full process model. The member of the MEC that achieved the highest likelihood was chosen, temporal links added to each variable, and parameters fit using maximum likelihood on the training set.

In this work, we extend the hillclimbing search to consider synchronic (temporal) dependencies in addition to asynchronic dependencies. We also allow for temporal edges between different sensor types (for example, the temperature process at time t can be conditioned on solar radiation at $t-1$). The extension is done by performing the aforementioned hillclimbing structure search over $\left\{ \mathbf{X}^t, \mathbf{X}^{t-1} \right\}$ with a set of edge constraints. The edge constraints specify those directed edges that *cannot* be considered in the hillclimbing steps that either add a new edge, or reverse an existing edge, between two variables. Specifically, we enforce that any temporal edge points forward in time or, more formally, that any edge between a variable X_i^{t-1} and X_j^t is such that $X_j^t | X_i^{t-1}$. Though the resultant structure also belongs to a MEC, we do not need to consider any other members of that MEC, because we do not append additional variables or edges (all variables were included in the initial structure learning). Parameters of the learned process model are again fit using maximum likelihood on the training set. The hillclimbing structure search is performed using 500 random restarts, and each variable is allowed a

maximum of $|\mathbf{X^t}|$ parents.

3.6.2 Evaluating the Benefit of Additional Sensors

We believe that the incorporation of additional sensors should lead to an overall better QC model. With more sensors, the model becomes robust to the failure of any single sensor, as it will be able to leverage correlated observations from the remaining working sensors in the network. Furthermore, the process component of the QC model can represent a richer set of interdependencies among the phenomena being measured by the sensor network, because more of those sensors can be included in the model. Consequently, we expect higher accuracy in identifying when sensor failures occur. However, we believe that approximating the asynchronic inference with the methods described in Section 3.4 may negatively impact the accuracy of our model. In this section, we explore the trade-off between the advantages gained by adding more sensors versus the cost (in accuracy) incurred by approximating inference.

To measure this trade-off, we learn 20 process models $P(\mathbf{X^t}, \mathbf{X^{t-1}})$, each covering a different number of sensors from $1, 2, \ldots, 20$. We then apply our approximate algorithms to these models and analyze their performance as a function of the size of the sensor network. Let us denote the number of sensors in the k^{th} process model as N_k. To determine which N_k of the 27 sensors are to be included in the process model, we first pick a target sensor $\mathcal{X}$. The target sensor is the same for all values of N_k; i.e., we use the same target sensor across all network sizes. Our evaluation of the inference algorithms will be with respect to this sensor. Next, we search among the 26 remaining sensors for the single sensor that provides the most *information* about $\mathcal{X}$, and then add it to the process model. More specifically, we are searching for the sensor that has the highest mutual information with $\mathcal{X}$:

$$\begin{aligned} \underset{X_i \in \mathbf{X} \setminus \mathcal{X}}{\operatorname{argmax}} I(\mathcal{X}; X_i) &= \underset{X_i}{\operatorname{argmax}} H(\mathcal{X}) - H(\mathcal{X}|X_i) \\ &= \underset{X_i}{\operatorname{argmax}} H(\mathcal{X}) + H(X_i) - H(\mathcal{X}, X_i). \end{aligned} \tag{3.14}$$

Here, the mutual information between $\mathcal{X}$ and X_i is evaluated using the observations o_i and $o_{\mathcal{X}}$ from those sensors in the datasets containing no data-anomalies, $F_1, \ldots, F_{10}$. The

entropy for a single sensor or set of sensors is calculated

$$H(X_i) = \frac{1}{2} \ln \left[(2\pi \exp)^N det(\hat{\Sigma}) \right],$$

where $\hat{\Sigma}$ is the sample variance of the observations from a single sensor X_i,

$$\bar{o}_i = \frac{1}{M} \sum_{m=0}^{M-1} o_i^m, \quad \hat{\Sigma} = \frac{1}{M} \sum_{m=0}^{M-1} (o_i^m - \bar{o}_i)^2.$$

If H is being calculated for a set of variables $H(\mathbf{X_i} \subset \mathbf{X})$, then $\hat{\Sigma}$ corresponds to the sample covariance of the multivariate Gaussian $P(\mathbf{X_i})$:

$$\hat{\Sigma} = \frac{1}{M} \sum_{m=0}^{M-1} (\vec{o_i}^m - \bar{\mathbf{o}}_i)(\vec{o_i}^m - \bar{\mathbf{o}}_i)^T.$$

For $N_k = 1$, the process model contains $\left\{\mathcal{X}^t, \mathcal{X}^{t-1}\right\}$. For $N_k = 2$, the process model contains $\left\{\mathcal{X}^t, \mathcal{X}^{t-1}\right\} \cup \left\{X_{1'}^t, X_{1'}^{t-1}\right\}$, where $X_{1'}$ denotes the result of the *argmax* in Equation (3.14). To construct a network of larger size N_k, we repeat the search in Equation (3.14); however, we condition the target sensor $\mathcal{X}$ on those sensors that were added when constructing the preceding, smaller networks: $X_{1'}, X_{2'}, \ldots, X_{k-1'}$. This implies that the network covering N_k sensors will cover the same set of sensors as the network having size $N_k - 1$, plus one additional sensor. The general form of this search is given as

$$\begin{aligned} &\underset{X_i \in \mathbf{X} \setminus \left\{\mathcal{X}, X_{1'}, \ldots, X_{k-1'}\right\}}{\operatorname{argmax}} I(\mathcal{X} | X_{1'}, \ldots, X_{k-1'}; X_i) = \\ &\underset{X_i \in \mathbf{X} \setminus \left\{\mathcal{X}, X_{1'}, \ldots, X_{k-1'}\right\}}{\operatorname{argmax}} H(X_{1'}, \ldots, X_{k-1'}, X_i) - H(\mathcal{X}, X_{1'}, \ldots, X_{k-1'}, X_i). \end{aligned} \tag{3.15}$$

For each network size $N_k = 1, \ldots, 20$, we train 10 process models using a combination of the clean-only data folds $F_1, \ldots, F_{10}$ described in Section 3.5. Let us denote the learned QC model $\mathcal{G}_{N_k}^f$ as the model having a process component $P(\mathbf{X^t}, \mathbf{X^{t-1}})$ spanning $2N_k$ sensors (N_k sensors at times t and $t-1$) and trained on fold set $\mathbf{F}_f$, where $\mathbf{F}_f = \mathbf{F} \setminus F_f$; i.e., $\mathbf{F}_f$ is the concatenation of observations from *all* training folds *except* the fold F_f. The QC model $\mathcal{G}_{N_k}^f$ is then evaluated using the noise-injected version of the hold-out fold, F_f'. We average the results of that evaluation across all 10 training folds/evaluation fold

combinations for a fixed network size N_k.

The metrics used for evaluations include accuracy, precision, recall, and mean-squared error (MSE). Accuracy, precision, and recall are a function of the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) described below.

- *True Positive* (TP): An anomalous observation o_i^t that the QC model correctly infers came from a broken sensor ($s_i^t = broken$).
- *False Positive* (FP): A clean observation o_i^t that the QC model erroneously infers came from a broken sensor ($s_i^t = broken$).
- *True Negative* (TN): A clean observation o_i^t that the QC model correctly infers came from a working sensor ($s_i^t = working$).
- *False Negative* (FN): An anomalous observation o_i^t that the QC model erroneously infers came from a working sensor ($s_i^t = working$).

Accuracy is computed $(TP+TN)/(TP+FP+TN+FN)$, and it is the probability that the QC model's classification of any given observation is correct. Precision, calculated $TP/(TP+FP)$, measures the probability that any observation labeled as a data-anomaly is actually an anomalous value. The recall metric, calculated as $TP/(TP+FN)$, reflects the percentage of all actual data-anomalies present in the dataset that are labeled as such by our method. Accuracy, precision, and recall are all rates measured on a $[0, 1.0]$ scale, where 1.0 is perfect performance. To calculate the MSE for sensor X_i in model $\mathcal{G}_{N_k}^f$, we measure the expected difference between the predicted value of X_i^t given observations from F_f', and the value of the observation o_i^t found in the clean version of the hold-out fold, F_f. Specifically, we calculate

$$MSE = \frac{1}{T}\sum_{t=0}^{T-1}(\mathbb{E}\left[P(X_i^t|\vec{o}^{\,0:t}, \vec{s}_{MAP}^{\,0:t})\right] - o_i^t)^2,$$

where $\vec{s}_{MAP}^{\,0:t}$ is the most-likely sensor configuration up to time t as computed by one of inference algorithms described in Section 3.4.

Figure 3.11 displays the results of our analysis for three different variations on the EP algorithm discussed in Section 3.4.2.1. Specifically, we examine the *Max-product* version that is equivalent to belief propagation (BPMAP-low), and the *iterative MAP* version

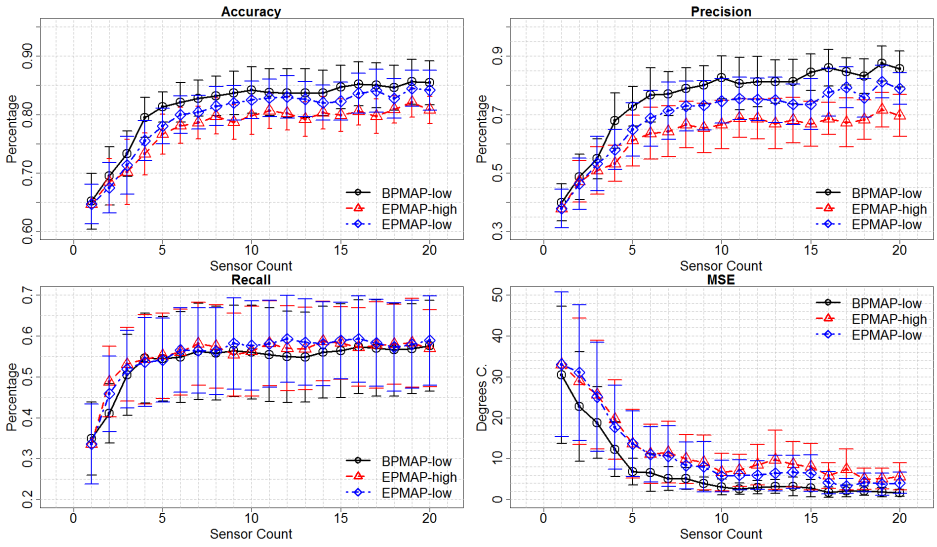

Figure 3.11: Performance as a function of N_k for three variants of the EP algorithm. The performance is evaluated in terms of accuracy (top left), precision (top right), recall (bottom left), and mean-squared error (bottom right). 95% confidence intervals are shown in addition to the metric score. The solid circle-line denotes the *Max-product* version of EP (BMMAP-low) with a low-to-high message-passing schedule, the dashed triangle-line denotes the *iterative MAP* algorithm with a high-to-low schedule, and dotted diamond-line denotes the *iterative MAP* algorithm with a low-to-high message passing schedule.

of EP (EPMAP-high and EPMAP-low). These algorithms are implemented using two different message-passing schedules. The "-low" suffix indicates a *low-to-high* schedule and "-high" indicates a *high-to-low* schedule. Results are not shown for the *marginal MAP* version of EP for the sake of clarity, and because they were near-equal to iterative MAP version. The target sensor $\mathcal{X}$ is the 1.5m air temperature sensor located at the Central meteorological station.

In general, all three inference algorithms show improvement in all metrics as the number of sensors increases from 1 to 10. The rate of improvement for each new sensor added lessens around 6 to 7 sensors, and eventually levels-off between 11 and 15 sensors. The accuracy and recall scores dip slightly in the interval of 11 to 15. For the first 10 values of N_k, the order in which sensors were added was nearly identical for all 10 trained models. At $N_k = 11$, the 2.5m or 3.5m Vanilla Leaf air temperature sensor were each

added in 3 of the models, with the other 4 models choosing a different unique sensor (see Figure 3.12). This may cause the results of the inference analysis to demonstrate more variation, because the process model structure is different for each of the $\mathcal{G}^f_{N_k=11}$ QC models. At $N_k = 15$, the accuracy, precision, and recall scores begin to improve as the models again become more homogeneous in terms of what sensors are included. Because some training folds contain instances of semi-rare events (for example, storms that cause rainfall, cloud coverage, and high winds), sensors that register these events (precipitation gauges, solar radiation sensors, and anemometers) may appear more informative than in the folds that do not contain such events. This could lead some models to incorporate these sensors during training, and then suffer a performance hit when the rare events do not occur in the corresponding test fold.

The *Max-Product MAP* (BPMAP-low) version outperformed both iterative versions of EP, EPMAP-low and EPMAP-high, in terms of accuracy, precision, and MSE. BPMAP's superior performance suggests that approximate message-passing does not work well in this domain, or that perhaps our constraint on messages having non-negative variance may be too restrictive. The former hypothesis is further supported by evidence that the "-low" confidence message-passing schedule works significantly better than the "-high" schedule. Recall that in the "low-to-high" schedule, we first fix the value of the sensor-state variable, S_i^t, that we are the most uncertain about. The collapsed Gaussian will look the least like either of its components when cases of high uncertainty occur, because the mixing weights of the Gaussian components will be closer to equal. In these cases, we achieve better results by picking one of the two Gaussian components in the mixture of Gaussians than by approximating the message with a collapsed Gaussian, as in Equation (3.13).

Recall scores appear approximately the same for all three EP variants for all numbers of sensors, though the confidence interval for these estimates are large. One reason for this may be that the synthetic anomalies follow a Markov process. If the QC algorithm classifies the first observation corrupted by noise as coming from a *working* sensor, then it will update the process model to reconcile the difference between $P(X_i^t)$ and o_i^t. This reconciliation is demonstrated by the process model "tracking" the anomalous observations for a period of time. For the *bias* anomaly type, observations r time-steps ahead of the first bias-injected value have a $.95^r$ chance of being similarly corrupted. If the model misclassifies the initial biased observation, then it will likely misclassify the remaining

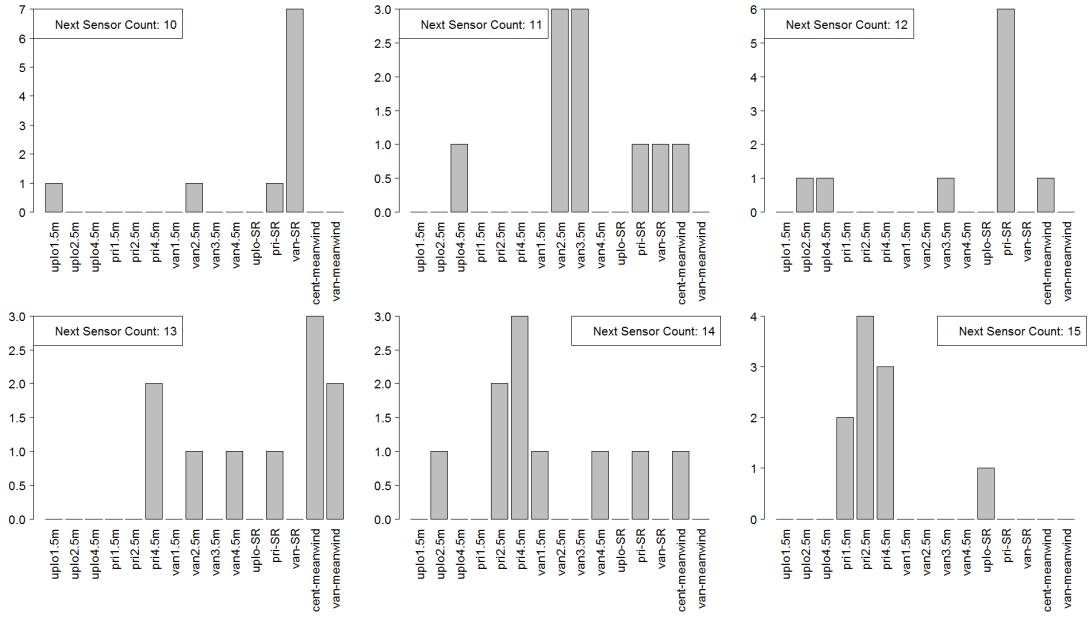

Figure 3.12: Each histogram above indicates how many of the 10 trained models (Y axis) chose one of above sensors (X axis) to add as the ${N_k}^{th}$ sensor for a model of size N_k. A separate histogram is shown for $N_k = 10, 11, \ldots, 15$. Prefixes "cent", "uplo", "pri", and "van" correspond to the station names: Central, Primary, Upper Lookout, and Vanilla Leaf, respectively. Suffixes "m", "SR", and "meanwind" correspond to sensor types: air temperature (thermometer), solar radiation, and anemometer. For example, for the 10^{th} sensor added for models of size $N_k = 10$, 7 of the 10 learned models chose the Vanilla Leaf solar radiation sensor (*van-SR*), 1 added the 1.5m Upper Lookout thermometer, 1 added the Vanilla Leaf 2.5m thermometer, and 1 added the Primary station's solar radiation sensor. Sensors not added by any of models between $N_k = 10$ and $N_k = 15$ (because they were either already included by all models or not selected by our greedy search) are not shown.

observations as coming from a *working* sensor. The result is that either the model correctly classifies the initial observation as a data-anomaly and then correctly labels the subsequent observations (higher TP count), or the QC model accepts the initial observation as *working* and labels the proceeding observations as non-anomalies (higher FN count).

In Figure 3.13, we compare the performance of the three approximate algorithms

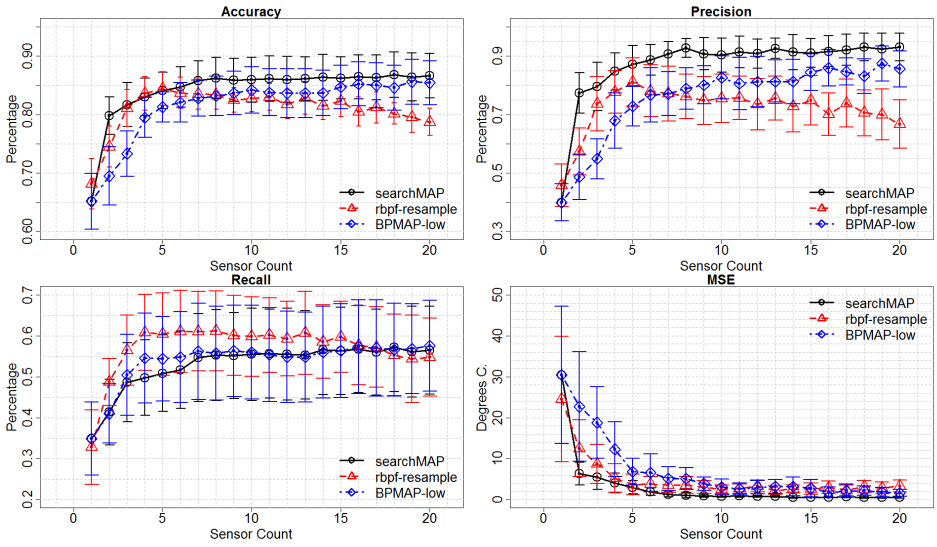

Figure 3.13: Performance as a function of N_k for the *Max-product* (BPMAP-low), Rao-blackwellized particle filter (rpbf-resample), and *SearchMAP* (SearchMAP) algorithms. The performance is evaluated in terms of accuracy (top left), precision (top right), recall (bottom left), and mean-squared error (bottom right). 95% confidence intervals are shown in addition to the metric score. The solid circle-line denotes the *SearchMAP* method, the dashed triangle-line denotes RBPF, and dotted diamond-line denotes the *Max-product* method (BPMAP-low) with a low-to-high message schedule.

described in Section 3.4. We have chosen *Max-product* to represent the EP-style approach, as it performed the best of the EP variants. Despite its simplicity, The *SearchMAP* method achieves the highest values of accuracy, precision, and MSE across almost all values of N_k. At $N_k = 20$, *SearchMAP* achieves a precision of .93, recall of .57, and mean-squared error of .52 ℃. The behavior of Rao-Blackwellized particle filter (*rbpf-resample*) appears counterintuitive, as its overall performance actually decreases as a function of the sensor count. After $N_k = 5$, *rbpf-resample*'s precision begins to decrease, as does its recall after $N_k = 12$. This is because we had to reduce the number of particles used as N_k grew in order to maintain a reasonable computational time for the inference query at each time t. The number of particles used for a network size N_k was $3250 - 100N_k$. However, we believe that had we held the number of particles fixed for all values of N_k, we would still expect to see a decrease in performance for sufficiently large N_k. The

space we need to sample (all possible configurations of N_k sensor-state variables) grows exponentially with N_k, and so we would expect our performance to suffer as the space grows and the particle count does not. While the computational cost grows linearly in the number of particles, the cost to score each particle under a process model of size N_k grows polynomially in N_k. We may be able to reduce the cost of scoring each particle by assuming a simpler proposal distribution (one that does not use a Kalman filter model); however, any computational benefit may be offset by the need to draw more particles.

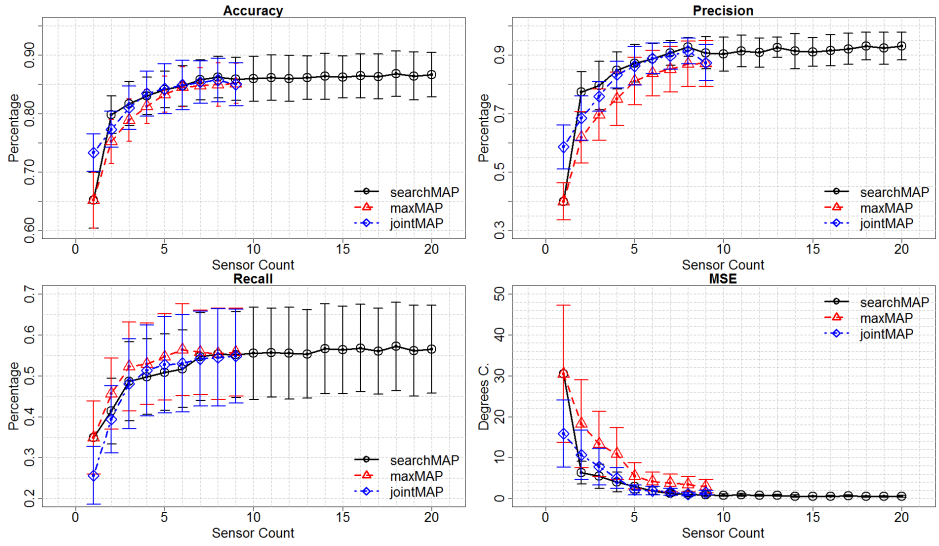

Figure 3.14: Performance as a function of N_k for the inference algorithms *SearchMAP*, *maxMAP*, and *jointMAP*. The performance is evaluated in terms in accuracy (top left), precision (top right), recall (bottom left), and mean-squared error (bottom right). 95% confidence intervals are shown in addition to the metric score. The solid circle-line denotes the *SearchMAP* method, the dashed triangle-line denotes *maxMAP*, and dotted diamond-line denotes the *jointMAP* algorithm.

We also compared our approximate algorithms to the exhaustive asynchronic algorithm used in our previous work and described in Section 3.3.2. This comparison was done to evaluate the performance difference of using approximate inference on a larger process model versus an exact (within a time slice) approach for computing the MAP on a smaller model. Figure 3.14 displays the results of this comparison using the approximate method that performed best, *SearchMAP*, and the method from our previous

work, *maxMAP* (described in Section 3.3.2). In addition, we include a third algorithm, *jointMAP*, which is a slight variation on *maxMAP*. Within a time slice, *jointMAP* computes Equation (3.1) identically to *maxMAP* (by exhaustively searching all 2^{N_k} sensor configurations and selecting the most probable configuration), however its solution to Equation (3.2) uses a collapsed Gaussian over all 2^{N_k} components instead of the highest-weighted component. In effect, *jointMAP* performs Gaussian summation over the 2^{N_K} components, and then collapses the resultant mixture onto a single Gaussian [1].

The *maxMAP* and *jointMAP* algorithms are only evaluated on networks of up to size $N_k = 9$, as the exponential cost grows intractable to compute for sizes $N_k \geq 10$. For a single sensor $N_k = 1$, the results of *maxMAP* is equal to *SearchMAP*, because both algorithms are behaving identically. Beyond $N_k = 9$, the improvements in accuracy, precision, recall, and MSE attained by *SearchMAP* are fairly small, though not completely negligible. The rapid onset of diminishing returns is, in part, due to the redundancy in air temperature sensors at the H.J. Andrews. The 1.5m air temperature sensor at the Central met. station has 3 air temperature sensors located a few meters higher on the station, which record observations nearly identical to its own. In all 10 training sets, the 2.5m and 4.5m thermometers at Central met. were added at $N_k = 2$ and $N_k = 3$, respectively. Thus, models only having a few sensors can do very well due to the high mutual information of these sensors with the target variable.

Within the range $2 \leq N_k \leq 9$, the greatest performance difference between *SearchMAP* and the other two algorithms is in precision and MSE. Somewhat surprisingly, the *SearchMAP* algorithm performs as well, if not better than, either exhaustive algorithm. We suspect this is because the *SearchMAP* algorithm is biased toward configurations of sensor-state variables that include fewer *broken* sensors. This bias manifests itself in a non-intuitive way. Consider the case where there are 2 correlated air temperature sensors that produce observations o_1^t and o_2^t. Further, suppose that the observations appear unusual compared to their measurements in the previous time step $\left\{o_1^{t-1}, o_2^{t-1}\right\}$. The exhaustive search algorithms (*maxMAP* and *jointMAP*) will evaluate all 4 configurations of *working*/*broken* and select the configuration that maximizes the likelihood of the observations. In this case, the all-*broken* configuration is the most likely, because the QC model has no apriori preference toward any specific configuration of sensor-states. *SearchMAP* will consider three configurations at first: $\{working, working\}$, $\{broken, working\}$, and, $\{working, broken\}$, and only evaluate the

4th configuration $\{broken, broken\}$ if the 2nd or 3rd configuration is more likely than the all-*working* configuration. Because setting only one of the sensors states to *broken* does not explain the observations any better than assuming both are *working*, *SearchMAP* chooses the all-*working* configuration. If the observations were actually valid (not data-anomalies), then the exhaustive algorithms would continue to misclassify subsequent observations as anomalies until the variance of the process model grows sufficiently large to explain the most recent observations.

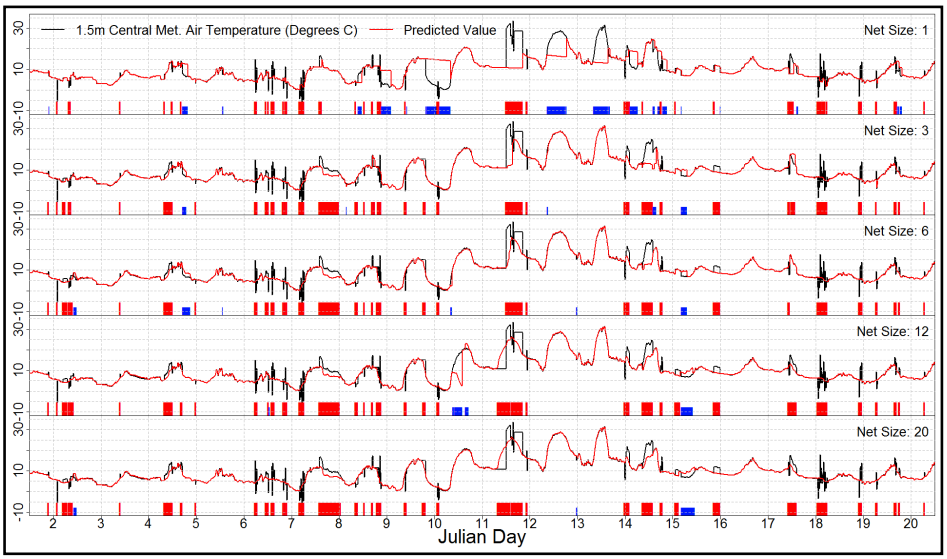

Figure 3.15: Time series plot showing the results of the *SearchMAP* algorithm using model $\mathcal{G}_{N_k}^{f=0}$ evaluated on test fold F_0'. The plot shows 21 days of quarter-hourly measurements from the 1.5m thermometer located at the Central station. Each plot contains results for a different number of included sensors: $N_k = 1, 3, 6, 12,$ and 20. In each plot, the black line corresponds to the noise-injected observations measured at the sensor. The red line indicates the imputation of the true value by the QC model. Tall red-hashes at the base of each plot indicate true positives, and short blue-hashes indicate false positives.

Figure 3.15 shows the results of the *SearchMAP* algorithm applied our synthetic dataset using networks of increasing size from $N_k = 1$ to $N_k = 20$. At $N_k = 1$, the process model is an AR1 (autoregressive, order-one [7]) model. It detects many of the true anomalous readings (true positives shown as large, red vertical hashes) as the larger network structures; however, it generates more false positives (short, blue vertical hashes)

and poorly imputes the "correct" value of the observation in cases where the sensor is *broken*. The poor imputation is especially evident on day 11, where the model's imputation is just the value of the last observation it believed to come from a *working* sensor (≈ 15 ℃), plus a small drift. At $N_k = 3$ and $N_k = 6$, the model has sufficient information to correctly classify steep changes in temperature (midnight betweens days 9 and 10, sunrise on days 12 and 13) as non-anomalous. It also successfully classifies the air temperature on day 11 as coming from a *broken* sensor, though it cannot yet correctly impute the shape of diurnal trend in air temperature. Instead, it classifies the flat line observations beginning 1/3 into day 11 as non-anomalous and does not register that the sensor is *broken* until a spike occurs approximately halfway through day 11. The addition of 6 more sensors ($N_k = 12$) allows the model to detect the injected noise on day 11 and correctly impute the affected values. Unfortunately, influence from one of new sensors convinces the QC model that the air temperature on day 10 is rising too quickly, and so it erroneously marks this period as anomalous, which incurs some false positives. This is resolved when the model has access to $N_k = 20$ sensors, at which point it produces very few false positives, except for a small period after day 15.

3.6.3 Complete Model: Synthetic Data

In an application setting, the goal of our methodology is to perform simultaneous quality control for *all* sensors in a given sensor network as opposed to a single target sensor. As such, we are interested in the performance of the model across all sensors. To evaluate this, we learned 10 separate process models $P(\mathbf{X}^t, \mathbf{X}^{t-1})$, with each model trained a different combination of data folds as described in Section 3.6.2. We then evaluated performance using the same metrics from the previous section, for all of the sensors. The models, $\mathcal{G}^f_{N_k=27}$ for $f = 1, \ldots, 10$, span all 27 sensors, and so no mutual information criteria was required to decide which sensors to add. Evaluation for each model was performed in same way as in Section 3.6.2: by testing on the noise-injected equivalent of the hold-out data for the QC model $\mathcal{G}^f_{N_k=27}$, referred to as F'_f. Results were then averaged across all 10 test folds.

Figure 3.16 (top) displays the precision and recall scores for 4 air temperature thermometers at the Primary station, as well as that station's solar radiation sensor, anemometer, and one of two precipitation gauges at the Upper Lookout site. We compared the per-

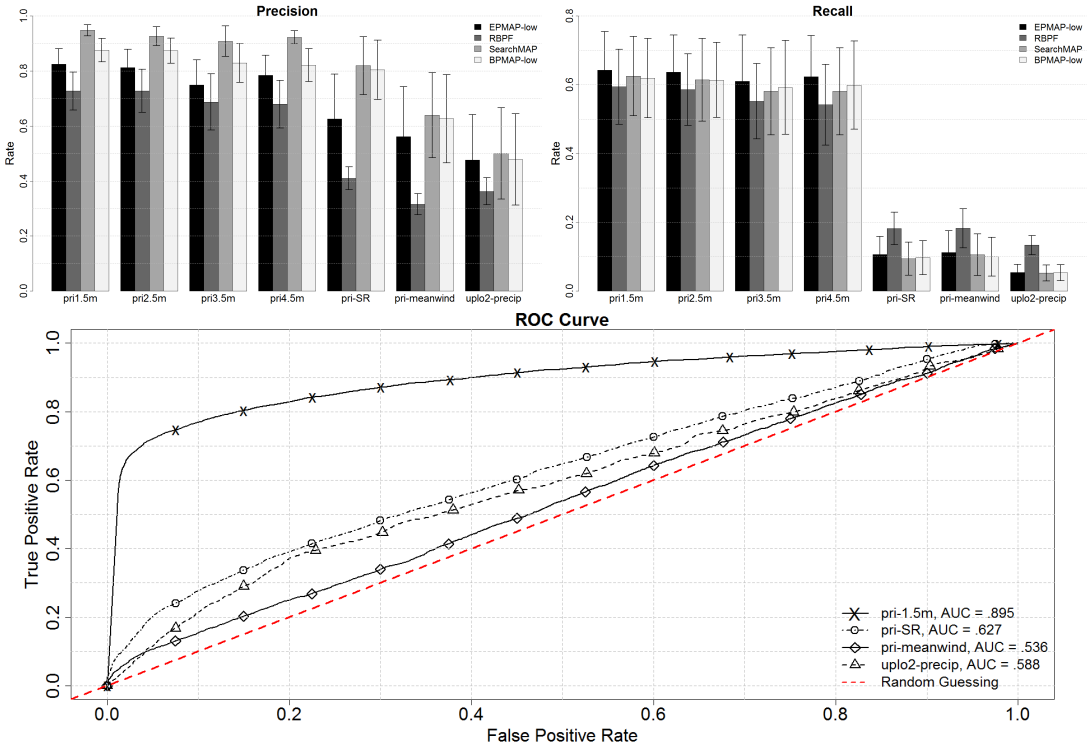

Figure 3.16: Top left: average recall scores on the synthetic data for 6 of 27 sensors included in the full QC model, plotted with 95% confidence intervals. Top right: average precision scores for the same sensors, dataset, and QC model. Bottom: ROC curve for 4 of the 27 sensors included in the full QC model.

formance of four different algorithms on the full QC model: EPMAP-low (*Iterative MAP* with a low-to-high schedule), RBPF (with resampling and 1000 particles), SearchMAP (our greedy hillclimbing inference algorithm), and BPMAP-low (*Max-Product MAP* with a low-to-high schedule). The SearchMAP algorithm achieved the highest precision scores across all types of sensors, though there is significant overlap in confidence intervals with BPMAP-low for the solar radiation, wind speed, and precipitation sensors. While precision scores were high for the air temperature thermometers ($> .90$), performance was generally poor for the other sensor types. We alluded to one possible cause for this in the previous section, in that there is much less redundancy for non-air-temperature sensor types at the HJ Andrews. Though each meteorological station is equipped with 4 ther-

mometers 1-meter apart on a tower, they typically only contain one of the other sensor types. A second reason may be that our process model cannot accurately capture the relationships among air temperature, solar radiation, precipitation, and mean wind using only linear functions of the raw observations. Recall scores were nearly-equal among all four inference algorithms applied to the full QC model; however, RBPF appeared to score better on the non-air-temperature sensors. On the solar radiation sensor and anemometers, it achieved $\approx .20$ recall compared to $\approx .10$ from the other 3 algorithms.

The bottom portion of Figure 3.16 shows the receiver operating characteristic (ROC) curves for 4 different sensors and the SearchMAP algorithm. To generate this curve, we computed posterior marginals at each of the sensor state variables in the following way. First, SearchMAP is used to compute the MAP sensor-state configuration at time t as described in Section 3.4.3; let us denote the computed MAP configuration at time t as $\vec{s}^{\,t}_{MAP}$. Next, to compute the marginal at each sensor-state variable S^t_i, we remove S^t_i's MAP value from $\vec{s}^{\,t}_{MAP}$ and compute the posterior $P(S^t_i|\vec{o}^{\,t}, \vec{s}^{\,t}_{MAP} \setminus s^t_i)$. We then generate the ROC curves in the usual way, by varying a classification threshold ζ, such that we declare $S^t_i = broken$ if $P(S^t_i = broken|\vec{o}^{\,t}, \vec{s}^{\,t}_{MAP} \setminus s^t_i) > \zeta$. Intuitively, we can think of adjusting ζ as a post-hoc approach to tuning the variance parameters of the observation model (σ^2_w and σ^2_b) and/or the prior probability on the working state of the sensor (p) to optimize classification performance.

Given the high precision scores shown in the top part of Figure 3.16, it is unsurprising we can trade-off this precision for higher recall. In the case of air temperature sensors, we can achieve .70 recall at 3.8% false positive rate (FPR, calculated as $FP/(FP+TN)$) on the synthetic noise data. What is more surprising is that our initial evaluation of precision and recall suggests the model does worse on precipitation sensors than anemometers. For values of recall (true positive rate, TPR) $< .10$, the ROC curve indicates that our classification is more accurate on the wind data than precipitation data; however, for recall rates $> .1$, performance on the precipiation sensor exceeds the windspeed sensor. In fact, the area under the curve (AUC) for the precipitation data is .05 greater than that of the windspeed sensor.

3.6.4 Complete Model: Real Data

We applied the same four inference algorithms from Section 3.6.3 to raw sensor observations collected from the H.J. Andrews. Here, we used only one QC model that included the 27 sensors from the H.J. Andrews network and averaged the results across 10 separate test sets. The process model was trained using clean-only data over a two-year period. The final training dataset consisted of approximately 31000 observations, as we could only use consecutive 15-minute periods that contained no data-anomalies in any of the 27 sensors. The model was then evaluated on 10 test sets, each of which contained a unique record of 14016 quarter-hour observations. The full test set corresponded to four years of sensor readings taken from the H.J. Andrews, exclusive of the 2 years selected for training. The results of our analysis are shown for a subset of those sensors in Figure 3.17.

Precision results for all four selected inference algorithms are low ($\approx .41$ or below); however, in the analysis of raw data, the generally poor precision results are unsurprising. Unlike in the synthetic dataset, the distribution over data-anomalies in the 10 test sets is significantly non-uniform. Many of the datasets contain very few examples of actual anomalous values. For example, 4 of the 10 test sets contain 0 anomalous observations from the 1.5m Primary met. station thermometer, 3 of the 10 have < 10 such values, and the other 3 have 64, 125, and 1331 anomalies. The average false positive rate of the SearchMAP algorithm for this particular sensor over the dataset is $.0017 \pm .0014$, or approximately 24 false-positives per test set. Thus, for 4 of the 10 test sets, we will achieve 0 precision because the algorithm identifies at least one observation as a data-anomaly when none are actually present. For the 3 data sets where the true anomaly count is less than 10, we will achieve precision scores $< .5$. It follows, then, that the *average* precision across the entire test set will be low. The SearchMAP algorithm does score higher in precision than EP, RBPF, or BP on average; however, it shows significant overlap with the other algorithms in its 95% confidence intervals. For the same reason our precision scores appear dismal on real data, the inference algorithms score high recall, on average, across the test folds. If a given test set has zero instances of data-anomalies for a specific sensor, then the recall rate on that set is 1.0. Still, all four algorithms successfully detect most data-anomalies present in the real data, with recall scores approximately equal (within 95% confidence bounds).

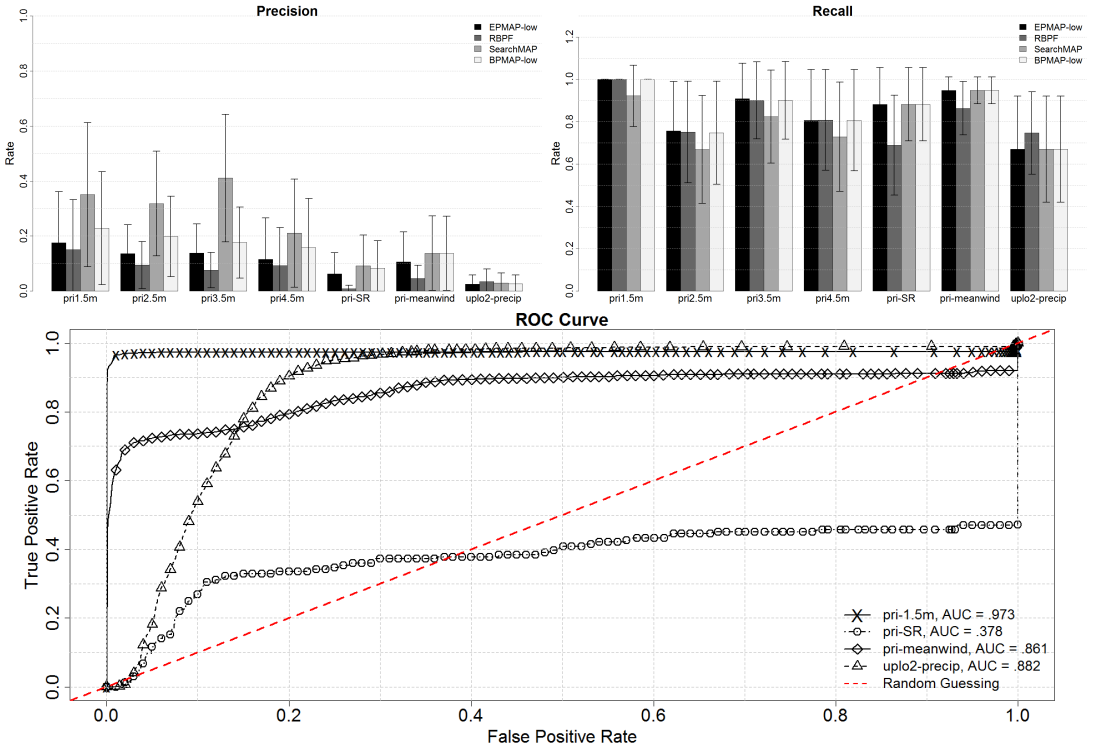

Figure 3.17: Top left: average recall scores on actual data shown for 6 of 27 sensors included in the full QC model, plotted with 95% confidence intervals over 10 test datasets. Top right: average precision scores for the same sensors, dataset, and QC model. Bottom: ROC curve for 4 of the 27 sensors included in the full QC model.

The ROC curve in Figure 3.17 (bottom) displays the recall rates of the SearchMAP algorithm as a function of its FPR on the real dataset. As noted above, because precision scores are not particularly meaningful on the real test sets, characterizing the recall trade-off in terms of FPR is more appropriate for site managers. In effect, the FPR indicates what percentage of the data would have to be manually inspected to remove all false positives from a *reduced* set of observations flagged by our QC method as anomalous. For example, if the manager wanted to remove at least 50% of anomalous preciptiation gauge readings from the Upper Lookout station, this could be done while achieving a 10% FPR. Across our 10 test sets, there were an average of 545 anomalous observations per test set at the Upper Lookout out precipitation gauge, which means that our QC

algorithm would classify $.1 * 14016 + .5 * 545 \approx 1674$ measurements as data-anomalies. The site manager would then have to manually inspect only this subset of values ($\approx 12\%$ of a test set) to remove all false positives. In some caes, the ROC curve peaks in recall (true positive rate) below 1.0 for all levels of FPR. This occurs because there are some observations labeled by the domain expert as anomalies that SearchMAP assigns very-low probability to coming from *broken* sensors. This early-peaking is particularly noticeable for the solar radiation sensor (pri-SR). For this sensor, the cumulative number of data-anomalies across all test fold is only 164, of which 82 come from the first test fold. These 82 anomalies manifest as small positive values of solar radiation being measured at night time. While the domain expert can be certain these sensors should measure no solar radiation at night, the QC model cannot signficantly distinguish values of 0 from small positive valies (i.e., 0.1). We note that this a limitation of linear-Gaussian representation among the process variables, and not any of the inference algorithms.

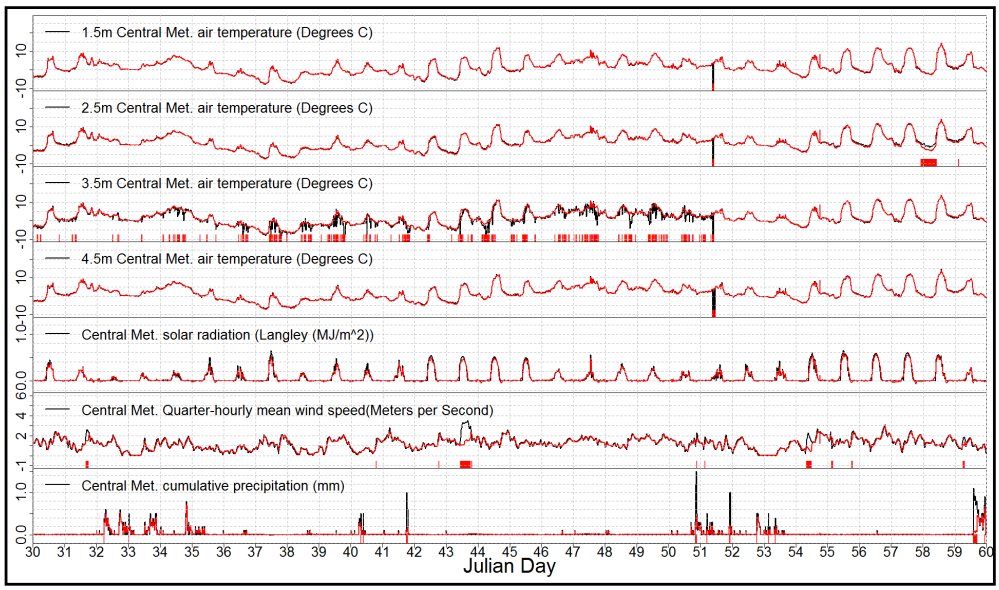

Figure 3.18: Results for the SearchMAP algorithm applied to the full QC model plotted for 7 sensors located on the Central benchmark meteorological tower, including 4 air temperature thermometers, a solar radiation sensor, anemometer, and precipitation gauge. The red line denotes the imputation of the sensor's value, and red hashes at the base of each plot indicate observations the model classified as coming from *broken* sensors.

In Figure 3.18, we display the results of the SearchMAP algorithm applied to the same segment of real data shown in Figure 3.10 with the 27-sensor model. The model/algorithm correctly flag the series of erratic sensor measurements from the 3.5m Central thermometer, in addition to the logger malfunction on day 51 affecting all four temperature sensors. A series of false positives occur in the 2.5m thermometer on day 58, as the air temperature at that sensors measures slightly higher than the QC model's estimate. The model raises only a few false positives for the anemometer, but fails to diagnose the frozen sensor on days 33 and 53. There are also some scatted false positives for the precipitation gauge, which are associated with large spikes in measured precipitation.

3.7 Related Work

Over the last several years, machine learning-based solutions to environmental sensor QC have increased in popularity. We describe some recent works that similarly employ probabilistic approaches to model uncertainty in the QC task. Osborne et al. [53] place a Gaussian process (GP) prior on the latent value of the process being tracked by a hydrological sensor. For a single sensor, the GP approach is similar to our own in that the prior distribution over the value of the latent process for t time steps follows a $t-$dimensional multivariate Normal distribution, with each dimension corresponding to a time slice. However, in the GP framework, the covariance matrix of the MVN is parameterized by a mean function μ and covariance function K. The general form of these functions allows for non-linear relationships in the temporal dependency structure, while our model requires the mean of $P(x^t)$ to be a linear function of x^{t-1}. Consequently, the authors can represent a more complex temporal relationship for a single sensor. Unlike in our model, the GP model has no spatial component that correlates observations from multiple sensors at the same time slice. Instead, the authors use a sliding window of fixed length over past observations, and then leverage the past sensor measurement to infer the value of the next observation. A consequence is that, if the window length is shorter than the longest consecutive period of breakage for a sensor, the GP will have little information to make a prediction.

Hill et al. [30] explore variations on the DBN representation to perform automated QC of meteorological data. Specifically, they compare a generic Kalman filter, a robust Kalman filter model where $P(\mathbf{O}|\mathbf{X})$ is modeled as a 2^N-component mixture of Gaussians

(as in Figure 3.3), and a switching linear dynamical system (SLDS) similar to our own model. In the case of the SLDS model, the authors do not explicitly factor the sensor-state variables or process variables; instead, at each time slice, there is a single 2^N-state discrete variable for the sensor state, and a N-dimensional MVG for the distribution of the latent process. For a sensor network containing 8 sensors, they achieve the best results with the SLDS model and RBPF for inference with 50000 particles. The authors note that their RBPF algorithm would require exponentially more particles for larger sensor networks to achieve consistent levels of performance. A static Bayesian network is used by Wang et al. [70] that incorporates spatial and temporal dependencies. For each sensor in the network, the authors instantiate an *AR-d* process, which models the current observation at the sensor as a linear function of its past d observations. Conditional dependencies among the sensors are only modeled at the most recent time step. There is no distinct sensor-state or observation variable for each sensor, and so the AR structure and use of Gaussian-only variables makes inference in this formulation efficient. Anomaly detection is performed by comparing the likelihood of observations from individual sensors using that sensor's AR model, $L_{\mathbf{y}_t} = L(\mathbf{y}_t|\theta_{B_\mathbf{y}})$, to a null hypothesis that all observations are valid. Similarly, the full set of sensor observations are evaluated with the previously described full model. Given a set of observations from a all of the sensors, the model classifies it belonging to one of 6 event types (*individual*, *composite*, *group*, etc.) determined by the independent and joint model likelihoods.

Relevant to the inference challenges posed by probabilistic QC models, Barber et al. address the issue of learning and detecting weather regimes that affect the correlation dynamics in wind sensors (measuring velocity) at multiple sites [4]. The authors employ a conditional linear-Gaussian HMM; however, the latent component of the model is the "regime" responsible generating observations at each of the wind sensors. This is analogous to our work where we have 2^N regimes with pre-specified dynamics (each *broken* sensor disconnects an observation from the process), whereas the authors are learning the regimes directly from the data. They handle inference in a means similar to the $maxMAP$ algorithm described in Section 3.3.2, but instead preserve the c most likely components in the forward message instead of the single most likely component.

The MUSCLES system, developed by Yi et al. [36], uses a multivariate-autoregressive model that adapts to changes in dynamics as the multivariate time series evolves. Anomaly detection is performed by comparing a held-out observation with the value imputed by

the model, given observations for the other $N-1$ signals for order-t time steps and the $t-1$ observations at the held-out signal. It follows that performing QC is relatively inexpensive (computationally), as the problem has been reduced to multiple-linear regression. The model adapts to changes in the relationship among the signals over time by incorporating a decay rate parameter that assigns low weight to regression weights associated with older observations.

In work by Platt et al. [56], the authors use a variational inference technique known as mean-field inference [34] to apply QC to a web service. In this domain, there are 10^5 observations associated potentially 10^4 fault cases at every time step. In their graphical model of the web service domain, they approximate the posterior probability distribution over a set of fault rates $\vec{F}$ given observations $\vec{V}$, which correspond to success/failures notices of client-server transactions. The form of this approximation makes the fault rates for different failure cases independent, such that the posterior approximation $Q(\vec{F}|\vec{V})$ is the product of series of beta distributions, which each correspond to a fault rate parameter. Designing a good approximation to the true posterior is non-trivial in many applications, as well the requisite minimization of its KL divergence to the true distribution; still, variational methods remain a potential avenue to consider in future work.

3.8 Conclusions & Future Work

In this article, we have considered three inference algorithms that address the exponential scaling cost associated with inference in our probabilistic QC model. In addition, we have presented a review of these algorithms and described practical issues of implementing each algorithm in the QC model. We analyzed the trade-off between the benefits of including additional sensors and the inaccuracies introduced by approximating asynchronic inference. Our results suggest that, though we can improve performance in the QC task by including additional sensors, diminishing returns have a significant role in determining the overall benefit of these sensors. One hypothesis is that the degree to which diminishing returns manifest is partly a function of the sensor redundancy at the domain site. Further, we have introduced a greedy-search technique for performing asynchronic inference. Results on both synthetic and real sensor data support suggest that our algorithm achieves superior precision and similar recall to two state-of-the-art algorithms. Lastly, we demonstrated the performance of our model across different types of

environmental sensor data, including air temperature, solar radiation, mean wind speed, and precipitation.

This work has opened up a host of potential avenues for future research. First, we have only discussed filtering techniques for performing real-time QC. If were we willing to delay classification of an observation at time t until time $t+k$, we could perform fixed-lag smoothing to obtain a more-informed estimate of the true working status of the sensors. Two of the algorithm we discussed, RBPF and EP, have known extensions that allow them to compute the smoothed estimate of the latent state [22, 27]. It is unclear the best way to extend the SearchMAP algorithm, though a naïve approach would be to extend the search domain to the joint sensor-state configuration over multiple time slices.

In our experiments, we noted that the results for precipitation, wind, and solar radiation sensors were generally worse than those for air temperature sensors. In the case of wind speed and precipitation, we suspect this is because the relationship among the other environmental sensors is nonlinear and the Gaussian assumption is wrong. Precipitation, for example, is strictly non-negative and heavily biased toward having 0 counts. A correction within the probabilistic framework would be to learn the nonlinear dynamics among these variable types and/or use more appropriate distributions for precipitation, mean wind speed, and solar radiation (such as a Gamma distribution). Similar to the work by Barber et al. [4], we may wish to learn multiple operating regimes that explain different spatiotemporal correlation dynamics in the data. For example, we could instantiate a latent variable that tracks the presence of storm systems, which may increase precipitation and wind, while lowering solar radiation and air temperature.

As noted in Section 3.7, we have not yet explored how variational methods for inference would handle in this domain. The presence of diminishing returns suggest that, for some sensors, only a subset of the other sensors in the network provide useful information for predicting their value. This may suggest that mean-field methods or other variational approaches (such as the Boyen-Koller algorithm [6]) could achieve decent results in both of our defined inference tasks. However, the challenge remains to discover a "good" factorization. There exists work in this area for discovering the degree of "separability" in DBN models that may be useful for learning such a factorization [23, 55] as it applies to the BK algorithm. We could also consider a belief-propagation approach, where messages correspond to localized beliefs at large clusters of correlated sensors. Obviously, a single Gaussian (as in our application of EP) would be too simple of an approximation of

that message form. We could, however, consider a non-parametric message form, like the one described in Sudderth et al. [65]. In order to evaluate how practical this approach would be, we would need to evaluate how well a mixture of multivariate Gaussians could model the joint distribution over the sensor-states and associated latent processes within a cluster of sensors.

Chapter 4: Conclusion

4.1 Conclusion

In the first manuscript, we described a Bayesian approach to learning a graphical structure that represented the spatial correlations among a set of environmental sensors. The structure was then expanded to include temporal dependencies to create a QC system based on the resulting spatiotemporal process model.. We applied our full QC algorithm to data collected from deployments that were part of the SenorScope project at the École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland. The spatiotemporal model performed best in the QC task compared to process models that ignored either temporal or spatial correlation or that assumed a fully-connected spatial component. Most importantly, our approach addressed one of the key shortcomings of our initial single-sensor QC system: in cases where a single sensor malfunctioned for an extended period, we were able to detect the broken state of the sensor and impute reasonable estimates for the affected values. This was achievable without the use of a baseline function or an extensive record of observations at each deployment. In addition, we demonstrated how the model could provide some insight into redundancy of the network; i.e., to determine which sensors were the most accurate predictors (most informative) for a held-out sensor. The inference algorithm in this work performed an exhaustive search for the most likely sensor state configuration out of a set of 2^N possible configurations. This limited our approach to networks of 10 or fewer sensors. However, this obstacle laid the groundwork for our next contribution.

Our second manuscript addressed the problem of scaling our QC approach to larger networks containing dozens of sensors. Specifically, we analyzed three approximate algorithms that each avoid enumerating all possible configurations to compute the most likely sensor state-state configuration within at a time step (asynchronic inference). We provided a review of each of these inference techniques, and described their application to our general probabilistic QC model. Our results demonstrated that, by incorporating observations from additional sensors into the QC model, we could increase the precision of the model in identifying anomalous values present in the data. Furthermore, the costs incurred by approximating asynchronic inference with these algorithms were outweighed by increases in performance from incorporating observations from additional sensors. However, diminishing returns were a factor in these gains, which suggests that there may be a practical limit to the increased performance achieved by the inclusion of more sensors.

We also established that, in the QC domain, our greedy-search algorithm SearchMAP outperformed both Rao-Blackwellized particle filtering and variants on the Expectation Propagation algorithms in terms of precision and mean-squared error. These results were consistent on both a synthetic dataset containing noise-injected data anomalies and the raw sensor data collected from the H.J. Andrews Experimental Forest.

The joint contributions from both of these works represent significant steps toward a site-adaptive, data-driven tool for quality control of environmental sensor data. Our research demonstrates that we can learn from existing data to build models of the latent processes tracked by these sensor networks. We can, in turn, apply these models to gap-fill measurements corrupted by broken sensors, and perform real-time automated QC. Further, we can scale these models to networks that include dozens of sensors that monitor different types of environmental variables (air temperature, precipitation, wind, etc.). While we are encouraged by the progress we have achieved in this dissertation, there are many challenging research questions that still must be solved to obtain an ideal QC system. In the next section, we explore some open questions that our work has raised and suggest some areas for future research.

4.2 Future Work

We consider two main directions for future research. For each direction, we explore some unresolved aspects of our research and provide some discussion of initial ideas and related work.

4.2.1 Learning and Representation

The current representation of the latent state $P(\mathbf{X}, \mathbf{S})$ follows a switching linear dynamical system model. The switching component, represented by the sensor-state variables $\mathbf{S}$, allows the QC model to ignore observations it believes are data anomalies so that they do not influence its belief about the process. However, the the process model itself is a Kalman filter that captures only one regime, or instance of spatiotemporal dynamics. Can we introduce latent factors that allow for multiple operating regimes (spatiotemporal models), and more importantly, can we learn the parameters of these regimes in an unsupervised setting?

In our experiments in Chapter 3, we noted that the results for precipitation, wind, and solar radiation sensors were generally worse than those for air temperature sensors. In the case of wind speed and precipitation, we suspect this is because the relationship among the other environmental sensors is nonlinear and the Gaussian assumption is wrong. Precipitation, for example, is strictly non-negative and heavily biased toward having 0 counts. There are two potential research directions that may address this inconsistency.

First, we can explore if the non-linear relationships can be sufficiently modeled using piecewise linearity conditioned on a latent factor. A single spatiotemporal model cannot represent the behavior of each regime and, as a result, the learned model (itself a multivariate Gaussian) will have a larger variance over each process's value (this is similar to how we approximate a Gaussian mixture with a single Gaussian). However, if we introduce a latent discrete factor to "switch" between the regimes, we can learn the parameters of multiple process models – one for each regime type. Work by Barber et al. does just this [4], and applies the resultant model to forecasting estimates of wind data. Relevant to one of our domains, colleagues at Oregon State University believe that the air circulation at the HJA also exhibits multiple regimes (temperature inversion, cold air drainage, etc.)[11]. Not only would a QC model that explicitly represents these regimes be expected to perform better at detecting/imputing anomalies, the parameters of these learned regimes could be of scientific value to ecologists.

Second, we could employ a process model that does not make a linear or Gaussian assumption and/or use more appropriate distributions for precipitation, mean wind speed, and solar radiation (such as a Gamma distribution). Regarding non-linear process models, some examples exist in the form of extended and unscented Kalman filters. However, these models are applied in domains where the process dynamics are nonlinear but known apriori, so they do not have to be estimated. In our setting, we assume the process dynamics to be unknown and estimate them from data. Currently, it is not clear how we would estimate the spatiotemporal structure without the linear-Gaussian constraint. Still, the inference techniques described in Chapter 4 do not explicitly rely on a Gaussian or linear relationship among the variables. EP can be generalized to any exponential family distribution that can be represented with a small set of moments; RBPF only requires that we can generate a valid proposal that can be efficiently sampled; and the SearchMAP algorithm requires that we can score observations against $P(\mathbf{O^t}|\mathbf{S^{0:t}})$ for a fixed value of $\mathbf{S^{0:t}}$.

Once we have learned a process model at a new site, can its structure be leveraged to speed up learning at new sites? We suspect that many sensor networks share commonalities in the microclimates they cover, the spatial layout of the sensors, and the types of environmental data they measure. This is especially true of top-down monitoring organizations like NEON, where the policy is to apply a unified suite of sensors at each network location. One approach may be to build a catalogue of learned process models, along with descriptions of the site where they were deployed, and then match a new site to one of the existing ones based on a site-similarity metric. This would require a way of measuring similarities between sights, and a way to exploit knowledg from existing structures in the catalog during structure learning. We can do this for linear Gaussian network structures (as described in Chapter 2), because the BGe metric allows the stipulation of a prior distribution in the form of a process model. An opportunity for future research is to determine a method that generalizes to non-linear or non-Gaussian process models.

Lastly, it would be interesting to perform an analysis similar to that in Chapter 4, in which we extend the Markov order of the process model and hold the number of sensors included in the network N fixed. In this way, we could determine the optimal auto-regressive order for the different types of environmental variables in our QC model. It may be the case that some variable types have delayed effects; for example, solar radiation at time t may affect temperature at time step $t + k$. We note that, for each order added to the model, we are essentially adding another N sensors to the QC model, and so the cost may quickly grow prohibitive even with our approximate algorithms. Still, such an experiment could help answer the question, "is it better to include $2N$ sensors in a network or to model an $AR(2)$ process for N sensors?" The answer may suggest an inflection point where, as a result of diminishing returns, the QC model would be achieve better performance by extending the Markovian lags rather than by adding more sensors.

4.2.2 Inference

Up to this point, we have framed the QC task as requiring real-time inference, which has justified our use of filtering methods to compute $argmax_{\mathbf{S}^t} P(\mathbf{S}^t|\mathbf{O}^{0:t})$. We hypothesize that we could achieve better performance if we waited until time $t + k$ to compute the smoothed MAP estimate $argmax_{\mathbf{S}^t} P(\mathbf{S}^t|\mathbf{O}^{0:t+k})$. Is there a sufficiently large value

of k such that diminishing returns have a result analogous to increasing N? What other inference algorithms might we consider that handle a non-linear, non-Gaussian representation of the latent process?

Relating to the first extension, if were we willing to delay classification of an observation at time t until time $t + k$, we could perform fixed-lag smoothing to obtain a more-informed estimate of the true working status of the sensors. The additional information would come from having sensor observations from the future ($t + 1$ to $t + k$), in addition to past observations (0 to t). Two of the algorithm we discussed, RBPF and EP, have known extensions that allow them to compute the smoothed estimate of the latent state [22, 27]. It is unclear how to extend the SearchMAP algorithm, though a naïve approach would be to extend the search domain to the joint sensor-state configuration over multiple time slices.

We have not yet explored how well variational methods for inference would work in this domain. Results from experiments both Chapters 3 and 4 show that, for some sensors, only a subset of the other sensors in the network provide useful information for predicting their value. This may suggest that mean-field methods or other variational approaches (such as the Boyen-Koller (BK) algorithm [6]) could achieve decent results in both of our defined inference tasks. However, the challenge remains to discover a "good" factorization. There exists work in this area for discovering the degree of "separability" in DBN models that may be useful for learning such a factorization [23, 55] as it applies to the BK algorithm. We could also consider a belief-propagation approach, where messages correspond to localized beliefs at large clusters of correlated sensors. Obviously, a single Gaussian (as in our application of EP) would be too simple of an approximation of that message form. We could, however, consider a non-parametric message form, like the one described in Sudderth et al. [65]. In order to evaluate how practical this approach would be, we would need to evaluate how well a mixture of multivariate Gaussians could model the joint distribution over the sensor states and the associated latent processes within a cluster of sensors.

Bibliography

[1] Daniel Alspach and Harold Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, AC-17(4):439 – 448, August 1972.

[2] Steen Andersson, David Madigan, and Michael D. Perlman. A characterization of markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25:505–541, 1995.

[3] H.B. Aradhye. Sensor fault detection, isolation, and accommodation using neural networks, fuzzy logick, and bayesian belief networks. Master's thesis, University of New Mexico, Albuquerque, NM, 1997.

[4] Chris Barber, Joseph Bockhorst, and Paul Roebber. Auto-regressive hmm inference with incomplete data for short-horizon wind forecasting. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 136–144. 2010.

[5] Barbara J. Benson, Barbara J. Bond, Michael P. Hamilton, Russell K. Monson, and Richard Han. *Frontiers in Ecology and the Environment*, 8(4):193–200, 2010.

[6] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, UAI'98, pages 33–42, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[7] Chris Chatfield. *Time-Series Forecasting*. Chapman & Hall/CRC, New York, NY, 2000.

[8] Jacob A. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46, 1960.

[9] Scott L. Collins, Luís M. A. Bettencourt, Aric Hagberg, Renee F. Brown, D. I. Moore, Greg Bonito, Kevin A. Delin, Shannon P. Jackson, David W. Johnson, Scott C. Burleigh, Richard R. Woodrow, and J. Michael McAuley. New opportunities in ecological sensing using wireless sensor networks. *Frontiers in Ecology*, 4:402–407, 2006.

[10] C. Daly and W. McKee. Meteorological data from benchmark stations at the Andrews Experimental Forest. Long-term ecologica research. Forest Science Data Bank, Corvallis, OR. Database. Available: http://andrewsforest.oregonstate.edu/data/abstract.cfm?dbcode=MS001, April 2012.

[11] Christopher Daly, Jonathan W. Smith , Joseph I. Smith, and Robert B. McKane. High-resolution spatial modeling of daily weather elements for a catchment in the Oregon Cascade Mountains, United States. *Journal of Applied Meteorology and Climatology*, 45:1565–1586, October 2007.

[12] Christopher Daly, K. Redmond, W. Gibson, M. Doggett, J. Smith, G. Taylor, P. Pasteris, and G. Johnson. Opportunities for improvements in the quality control of climate observations. In *15th AMS Conference on Applied Climatology*, Savannah, GA, June 2005. American Meteorological Society.

[13] Kaustav Das and Jeff Schneider. Detecting anomalous records in categorical datasets. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 220–229, New York, NY, USA, 2007. ACM.

[14] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 524–529, Cambridge, Massachusetts, 1988. MIT Press.

[15] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In E. Horvitz and F. Jensen, editors, *Twelthth Conf. on Uncertainty in Artificial Intelligence*, pages 211–219, Portland, Oregon, 1996. Morgan Kaufmann.

[16] Ethan Dereszynski. A probabilistic model for anomaly detection in remote sensor streams. Master's thesis, Oregon State University, Corvallis, OR, 2007.

[17] Ethan Dereszynski and Thomas Dietterich. A probabilistic model for anomaly detection in remote sensor data streams. In Ronald Parr and Linda van der Gaag, editors, *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI 2007)*, pages 75–82, Vancouver, B.C., July 2007. AUAI Press.

[18] Ethan W. Dereszynski and Thomas G. Dietterich. Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns. *ACM Transactions on Sensor Networks*, 8(1):3:1–3:36, 2011.

[19] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised filtering for dynamic bayesian networks. In *Proceedings of the 16th*

Annual Conference on Uncertainty in Artificial Intelligence (UAI-00), pages 176–1, San Francisco, CA, 2000. Morgan Kaufmann.

[20] Arnaud Doucet, Neil J. Gordon, and Vikram Krishnamurthy. Particle filters for state estimation of jump markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, 2001.

[21] Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In Pat Langley, editor, *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 255–262, Stanford, CA, June 2000. Morgan Kaufmann.

[22] W. Fong, S.J. Godsill, A. Doucet, and M. West. Monte carlo smoothing with application to audio signal enhancement. *Signal Processing, IEEE Transactions on*, 50(2):438 –449, February 2002.

[23] Charlie Frogner and Avi Pfeffer. Discovering weakly-interacting factors in a complex stochastic process. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 481–488. MIT Press, Cambridge, MA, 2008.

[24] Dan Geiger and David Heckerman. Learning Gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research, Redmond, WA, 1994.

[25] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(20):721–741, 1984.

[26] Steven B. Gillispie and Michael D. Perlman. The size distribution for markov equivalence classes of acyclic digraph models. *Artificial Intelligence*, 141(1-2):137 – 155, 2002.

[27] Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic bayesian networks. In *In Proceedings UAI*, pages 216–223, 2002.

[28] David J. Hill and Barbara S. Minsker. Automated fault detection for in-situ environmental sensors. In *7th International Conference on Hydroinformatics*, Nice, France, 2006. Research Publishing.

[29] David J. Hill, Barbara S. Minsker, and Eyal Amir. Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the 32nd conference of IAHR*, Venice, Italy, 2007. International Association of Hydraulic Engineering and Research, IAHR.

[30] David J. Hill, Barbara S. Minsker, and Eyal Amir. Real-time Bayesian anomaly detection in streaming environmental data. *Water Resources Research*, 45(W00D28):1–16, March 2009.

[31] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, 2004.

[32] P.H. Ibarguengoytia, L.E. Sucar, and S. Vadera. A probabilistic model for sensor validation. In E. Horvitz and F. Jensen, editors, *Twelthth Conference on Uncertainty in Artificial Intelligence*, pages 332–333, Portland, Oregon, 1996. Morgan Kaufmann.

[33] Rolf Isermann. Model-based fault detection and diagnosis: Status and applications. In *Annual Reviews in Control*, volume 29, pages 71–85. Pergamon Press Ltd., St. Petersburg, Russia, 2005.

[34] Michael I. Jordan. An introduction to variational methods for graphical models. In *Machine Learning*, pages 183–233. MIT Press, 1999.

[35] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[36] Byoung kee Yi, N. D. Sidiropoulos, Theodore Johnson, and H. V. Jagadish. Online data mining for co-evolving time sequences. In *In Proceedings of the 16th International Conference on Data Engineering*, pages 13–22, 2000.

[37] Weng keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In Kenneth Ford, editor, *In Proceedings of the 18th National Conference on Artificial Intelligence*, pages 217–223, Edmonton, Alberta, 2002. AAAI Press.

[38] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[39] S.L. Lauritzen. Propogation of probabilities, means, and variance in mixed graphical association models. *Journal of The American Statistical Association*, 87(420):1098–1108, 1992.

[40] S.L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57, 1989.

[41] Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, Inc., New York, New York, 1996.

[42] Uri Lerner. *Hybrid Bayesian Networks for Reasoning About Complex Systems*. PhD thesis, Stanford University, Palo Alto, California, 2002.

[43] Uri Lerner and Ronald Parr. Inference in hybrid networks: Theoretical limits and practical algorithms. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 310–318, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[44] G. Matheron. Principles of geostatistics. *Economic Geology*, 53(8):1246–1266, December 1963.

[45] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141-3 of *Mathematics in Science and Engineering*. Academic Press, 1982.

[46] N. Mehranbod, M. Soroush, M. Piovos, and B. A. Ogunnaike. Probabilistic model for sensor fault detection and identification. *AIChe Journal*, 49(7):1787–1802, 2003.

[47] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[48] Thomas P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, January 2001.

[49] M. Mourad and J.L. Bertrand-Krajewski. A method for automatic validation of long time series of data in urban hydrology. *Water Science & Technology*, 45(4-5):263–270, 2002.

[50] Kevin Murphy. Bayesian map learning in dynamic environments. In *In Neural Info. Proc. Systems (NIPS)*, pages 1015–1021. MIT Press, 2000.

[51] Kevin P. Murphy. Inference and learning in hybrid Bayesian networks. Technical Report UCB/CSD-98-990, University of California, Berkeley, California, January 1998.

[52] A. E. Nicholson and J. M. Brady. Sensor validation using dynamic belief networks. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 207–214, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[53] Michael A. Osborne, Roman Garnett, Kevin Swersky, and Nando de Freitas. Prediction and fault detection of environmental signals with uncharacterised faults. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012.

[54] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[55] Avi Pfeffer. Approximate separability for weak interaction in dynamic systems. In *Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pages 375–384, Arlington, Virginia, 2006. AUAI Press.

[56] John Platt, Emre Kiciman, and David Maltz. Fast variational inference for large-scale internet diagnosis. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1169–1176. MIT Press, Cambridge, MA, 2008.

[57] John Porter, Peter Arzberger, Hans-Werner Braun, Pablo Bryant, Stuart Gage, Todd Hansen, Paul Hanson, Chau-Chin Lin, Fang-Pang Lin, Timothy Kratz, William Michener, Sedra Shapiro, and Thomas Williams. Wireless Sensor Networks for Ecology. *BioScience*, 55(7), 2005.

[58] John H Porter, Eric Nagy, Timothy K Kratz, Paul Hanson, Scott L Collins, and Peter Arzberger. New eyes on the world: Advanced sensors for ecology. *BioScience*, 59(5):385–397, 2009.

[59] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[60] Ben Y. Reis, Marcello Pagano, and Kenneth D. Mandl. Using temporal context to improve biosurveillance. *Proceedings of the National Academy of Science*, 100(4):1961–1965, 2003.

[61] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., Upper Saddle River, New Jersey, 2003.

[62] Mark Schmidt, Alexandru Nicolescu-Mizil, and Kevin Murphy. Learning graphical model structure using L1-regularization paths. In *Proceedings of the Twenty-Seceond AAAI Conference on Artificial Intelligence*, pages 1278–1284, Vancouver, British Columbia, 2007. AAAI Press.

[63] Sensirion, Sensirion AG, Laubisrütistr. 50, CH-8712 Stäfa ZH, Switzerland. *SHT1x / SHT7x Humidity & Temperature Sensor*, May 2005.

[64] W.M. Jr. Sheldon. Dynamic, rule-based quality control framework for real-time sensor data. In C. Gries and M.B. Jones, editors, *Proceedings of the Environmental Information Management Conference 2008 (EIM 2008)*, pages 145–150, 2008.

[65] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *Computer Vision and Pattern Recognition*, 2003.

[66] A. Szalay and J. Gray. The world-wide telescope, an archetype for online science. Technical Report MSR-TR-2002-75, MSR, 2002.

[67] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, 65(1):31–78, 2006.

[68] Anthony J. Viera and Joanne M. Garrett. Understanding interobserver agreement: The kappa statistic. *Family Medicine*, 37(5):360–363, 2005.

[69] Ling Wang, Marco F. Ramoni, Kenneth D. Mandl, and Paola Sebastiani. Factors affecting automated syndromic surveillance. *Artificial Intelligence in Medicine*, 34(3):269–278, 2005.

[70] X. Rosalind Wang, Joseph T. Lizier, Oliver Obst, Mikhail Prokopenko, and Peter Wang. Spatiotemporal anomaly detection in gas monitoring sensor networks. In *Proceedings of the 5th European conference on Wireless sensor networks*, EWSN'08, pages 90–105, Berlin, Heidelberg, 2008. Springer-Verlag.

[71] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report 95-041, University of North Carolina at Chapel Hill, Chapel Hill, NC, 2006.

[72] Ming Yuan and Yi Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.