Worst-case drift detection of sensor networks: performances and algorithms

Alexandre Reiffers-Masson

Worst-case drift detection of sensor networks: performances and algorithms

Alexandre Reiffers-Masson
RBCCPS, Indian Institute of Science, Bangalore
IMT Atlantique, Brest
Email: alexandre.reiffers-masson@imt-atlantique.fr

Abstract

The goal of this paper is to derive algorithms that are able to detect unreliable/drifted sensors, in a sensor network. To recover the state of each sensor, we restrict ourselves to a particular decoder, inspired by graph partitioning problems. We provide necessary and sufficient conditions over the measurements such that the decoder perfectly recovers each sensor binary state. The outputs of the decoder can be computed using a dynamic programming approach. One challenging part of this approach is the complexity of the dynamic programming equation. Indeed, the resolution time will increase exponentially with the number of sensors. Therefore, we propose an efficient heuristic method that approximately solves the problem. We study the performance of our algorithm using simulations.

I. Introduction

In a low-cost sensor network, sensors can gradually become unreliable. This phenomenon is called drift. For instance, it can arise either due to the loss of sensitivity (ex: electrochemical sensors) or aging of the material after long exposure to meteorological conditions. Such phenomena are observed in low-cost pollution sensors. In this paper, the task that we are trying to solve is the detection of drifted sensors when only the measurements of the sensor network are available.

Multiple papers have been trying to solve this problem and we describe the most relevant ones, due to the lack of space. Most drift estimation techniques depend on probabilistic filtering methods. For example in [6] and [4] the authors propose a method that interpolates ground truth based on spatial interpolation techniques such as kriging. Drifts were estimated based on the differences in the measured sensor readings and the kriging prediction, which were filtered using

a Kalman filter. However, the accuracy of these two methods is limited by the fact that they accumulate errors over time. We take a different approach, and solve a clustering problem by exploiting the band limited nature of the sensing field, as in [7], and we treat degraded sensors as outliers in our clusters as in [3] and [2].

In this paper, our contributions are the following: First, we introduce a particular decoder designed to detect drifted sensors. This decoder is the solution of a temporal graph partitioning problem. Secondly, we provide necessary and sufficient conditions over the measurements such that our decoder perfectly recovers the state of each sensor. Thirdly, we propose an efficient heuristic method that approximately solves the temporal graph partitioning problem.

II. SETUP

1) Time series model: Consider a network of I sensors that measures a finite set of spatially and temporally varying quantities. The set of sensors is denoted by $\mathcal{I} := \{1, \dots, I\}$. Let $x_i^n \in \mathbb{R}$ be the output of the sensor $i \in \mathcal{I}$, at a particular instant $n \in \{1, \dots, N\}$. We assume that there are C underlying processes. We denote by s_c^n one of such process, with $c \in \mathcal{C} := \{1, \dots, C\}$. At each instant n, we consider a scenario where the set of sensors can be partitioned into two sets $\mathcal{I}_1(n)$ and $\mathcal{I}_2(n)$ such that $\mathcal{I}_1(n) \cap \mathcal{I}_2(n) = \emptyset$ and $\mathcal{I}_1(n) \cup \mathcal{I}_2(n) = \mathcal{I}$. At each instant n, $\mathcal{I}_1(n)$ is the set of reliable sensors and $\mathcal{I}_2(n)$ is the set of drifted sensors. We assume that once a sensor drifts, it is unreliable for the remaining time. The latter can be translated into the following condition, $\mathcal{I}_2(n) \subseteq \mathcal{I}_2(n+1)$. At every instant n, the number of unreliable sensors is smaller that the number of reliable sensors, i.e. $\mid \mathcal{I}_1(n) \mid > \mid \mathcal{I}_2(n) \mid$, where $\mid A \mid$ denote the cardinality of a set A. We assume that a sensor measures only one underlying process and we consider a linear dynamical model as shown below:

$$\begin{aligned} x_i^n &= s_{c(i)}^n + \epsilon_i^n, \quad \text{if } i \in \mathcal{I}_1(n), \\ x_i^n &= s_{c(i)}^n + \epsilon_i^n + \delta_i^n, \quad \text{if } i \in \mathcal{I}_2(n), \end{aligned} \tag{1}$$

where for each $i \in \mathcal{I}$, $c(i) \in \mathcal{C}$ denote the underlying process measured by sensor i, $\epsilon_i^n \in \mathbb{R}$ denote the noise of sensor i at instant n. For $i \in \mathcal{I}_2(n)$, $\delta_i^n \in \mathbb{R}$ captures the drift value for sensor i at time n.

2) Noise and drift models: At every instant n and for every sensor i, we assume that the noise is such that $(\epsilon_i^n)^2 \leq c_\epsilon$. Moreover, we assume that if $i \in \mathcal{I}_2(n)$, then $\underline{c}_\delta \leq (\delta_i^n)^2 \leq \overline{c}_\delta$. We restrict our attention to deterministic approaches for the performance analysis. Indeed, we want to understand the performance of the drift detection task in worst-case scenarios. More precisely,

what are the conditions on c_ϵ, $\underline{c}_\delta$ and $\overline{c}_\delta$ such that the drifts can be detected and the time series can be filtered/recovered. Such approaches have been proven to be related to some specific noises model (Gaussian noise models for instance, see [1]).

3) Side information: Throughout this paper, we assume that, for every $(c, i) \in \mathcal{C} \times \mathcal{I}$, the underlying process c measured by sensor i is *unknown* (therefore for every i, $c(i)$ is also *unknown*). However, we assume that it is possible to access a *similarity matrix* $\mathbf{E} := [[e_{ij}]]_{1\leq i,j\leq I}$ which captures the similarity between sensors. For instance, if past reliable measurements $\mathbf{x}^{-\mathbf{N}:\mathbf{0}} := [[x_i^n]]_{1\leq i,j\leq I,-N\leq n\leq 0}$ are available, in the sense that for every $n \in \{-N, \ldots, 0\}$, $\mathcal{I}_2(n) = \emptyset$, then it is possible to construct $\mathbf{E}$ using the Radial Basis Function ($e_{ij} = \exp(-\|\mathbf{x_i}^{-\mathbf{N}:\mathbf{0}} - \mathbf{x_j}^{-\mathbf{N}:\mathbf{0}}\|^2/\sigma)$). If the similarity matrix is able to recover the entire underlying processes, then $e_{ij} = 1_{c(i)=c(j)}$. We will assume a less restrictive assumption, which is $\mathbf{E} = \mathbf{E}^* + \mathbf{M_E}$ where; (1) $\mathbf{E}^* = [[e^*_{ij}]]_{1\leq i,j\leq I}$ with $e^*_{ij} = 1_{c(i)=c(j)}$, (2) $\mathbf{M_E} \in \mathcal{M}_{I\times I}(\mathbb{R})$, such that $\max_{ij}\{M_{Eij}\} \leq c_{M_E}$. We assume that $\mathbf{E}$ is a symmetric matrix.

III. Drifts detection: Approach and Feasibility

A. Approach

The task studied in this paper is the following: How to recover $\overline{p}_i^n := 1_{i\in\mathcal{I}_1(n)}$ for every sensor i and every instant n knowing $\mathbf{x}^{\mathbf{1}:\mathbf{N}} := [[x_i^n]]_{1\leq i,j\leq I,1\leq n\leq N}$ and E? To solve such challenge, we reformulate our task as the problem of recovering binary sensors states from noisy edge measurements in a temporal graph. The definition of our temporal graph is as follows: the set of sensors $\mathcal{I}$ are the nodes of the graph; and an edge at instant n between sensor i and sensor j is given by $w_{ij}^n = (x_i^n - x_j^n)^2$. Depending on which set ($\mathcal{I}_1(n)$ or $\mathcal{I}_2(n)$) the node i and the node j are belonging to, the edge w_{ij}^n will have different values and therefore will reveal different information about the state of node i and node j:

$$w_{ij}^n = \begin{cases} (\epsilon_i^n - \epsilon_j^n)^2, \text{ if } (i,j) \in (\mathcal{I}_1(n))^2, \\ \\ (\epsilon_i^n - \epsilon_j^n)^2 + (\delta_i^n)^2 \\ \quad +2\delta_i^n(\epsilon_i^n - \epsilon_j^n), \text{ if } i \in \mathcal{I}_2(n),\ j \in \mathcal{I}_1(n), \\ \\ (\epsilon_i^n - \epsilon_j^n)^2 + (\delta_i^n - \delta_j^n)^2 \\ \quad +2(\delta_i^n - \delta_j^n)(\epsilon_i^n - \epsilon_j^n), \text{ if } (i,j) \in (\mathcal{I}_2(n))^2. \end{cases} \tag{2}$$

Having observed a set of matrices $\{W^n : 1 \leq n \leq N\}$ and an adjacency matrix E, the receiver uses a decoder $D : (\mathcal{M}_{I\times I}(\mathbb{R}_+))^N \times \mathcal{M}_{I\times I}(\mathbb{R}_+) \to \{1,\dots,N\}^I$, where the decoder $D(\{W^n : 1 \leq n \leq N\}, E)$ is returning a vector $\mathbf{m} := [m_i]_{1\leq i\leq I}$. For each sensor i, $m_i + 1$ is the predicted time at which sensor i starts to drift. We say that the decoder D is able to detect the drifts in the sensor network if $D(\{W^n : 1 \leq n \leq N\}, E) = [\sum_{n=1}^{N} 1_{\overline{p}_i^n=1}]_{1\leq i\leq I} =: \mathbf{m}^*$. In this paper, we will restrict to a particular decoder, inspired from graph partitioning problems. Consider the decoder D_λ such that $D_\lambda(\{W^n : 1 \leq n \leq N\}, E)$ is the optimal solution of the following integer programming problem:

$$\min_{\mathbf{m}\in\{1,\dots,N\}^I} C(\mathbf{m}) := \sum_{n=1}^{N}\sum_{i=1}^{I}\sum_{j=1}^{I} e_{ij} 1_{m_i\geq n} 1_{m_j\geq n}(w_{ij}^n - \lambda). \tag{3}$$

One important observation concerning the optimization problem (3) is that each of its local minimums (denoted by $\mathbf{m}^{\text{loc}}$) will satisfy the following local optimality condition:

$$\sum_{n=1}^{m_i^{\text{loc}}}\sum_{j=1}^{I} e_{ij} 1_{m_j^{\text{loc}}\geq n}(w_{ij}^n - \lambda) \leq \sum_{n=1}^{m'}\sum_{j=1}^{I} e_{ij} 1_{m_j^{\text{loc}}\geq n}(w_{ij}^n - \lambda),\ \forall i \in \mathcal{I},\ \forall m' \neq m_i^{\text{loc}}. \tag{4}$$

The term $e_{ji} 1_{m_j^{\text{loc}}\geq n}(w_{ji}^n - \lambda)$ does not appear in (4), because the matrix $\mathbf{E}$ and the matrix $\mathbf{W}^n$ are symmetric matrices.

Our decoder has an elegant interpretation in terms of graph partitioning. For every instant n and for a sensor set $\mathcal{J}^n \subseteq \mathcal{I}$, we define the weight of the set $\mathcal{J}^n$ by $\Omega^n(\mathcal{J}^n) = \sum_{(i,j)\in\mathcal{J}^n} e_{ij}(w_{ij}^n - \lambda)$. The optimization problem defined in (3) is equivalent to:

$$\min_{\{\mathcal{J}^n\subseteq\mathcal{I};\ n\in\{1,\dots,N\}\}} \sum_{n=1}^{N} \Omega^n(\mathcal{J}^n),\ \text{s.t. } \mathcal{J}^{n+1} \subseteq \mathcal{J}^n,\ \forall n < N.$$

This optimization problem is about finding the sequence of sets of sensors $\mathcal{J}^n$ such that the sum of weights $e_{ij}(w_{ij}^n - \lambda)$ inside $\mathcal{J}^n$ is as small as possible. Therefore we expect the decoder to decide that the sensors with the smallest w_{ij}^n will be inside the set $\mathcal{J}^n$. The constraint $\mathcal{J}^{n+1} \subseteq \mathcal{J}^n$ ensures that once a sensor is removed from the set $\mathcal{J}^n$, it should be removed forever. The goal of this constraint is to introduce robustness and to ensure that the decoder is considering a sensor as an unreliable sensor at time n if in the future ($n' > n$) the decoder will also take the same decision.

B. Going back to the time series model

In the rest of the paper, we will derive assumptions and algorithms for the general problem of recovering binary sensors states from noisy edge measurements in a temporal graph. But as an

exercise, we show how easy it is to go from assumptions over the matrices $\mathbf{W}^n$ to assumptions over the time series.

If we are going back to the time series model described in section II, our approach can be understood as follows. Let us assume that $\underline{c}_\delta - 2\sqrt{c_\epsilon \overline{c}_\delta} > 2c_\epsilon$. Then by taking $\underline{c}_\delta - 2\sqrt{c_\epsilon \overline{c}_\delta} > \lambda > 2c_\epsilon$, we will have $w_{ij}^n - \lambda < 0$, for every $(i,j) \in (\mathcal{I}_1(n))^2$ and $w_{ij}^n - \lambda > 0$, for every $(i,j) \in \mathcal{I}_1(n) \times \mathcal{I}_2(n)$. In this case the only global optimum of (3) is $\mathbf{m}^*$. But we believe that this assumption is too strong and in the next section, we derive necessary and sufficient conditions for perfect recovery when it exists $(i,j) \in (\mathcal{I}_1(n))^2$ such that $w_{ij}^n > \lambda$ and $(i,j) \in \mathcal{I}_1(n) \times \mathcal{I}_2(n)$ such that $w_{ij}^n > \lambda$.

For the rest of the paper, all our results will be over the matrices $\mathbf{W}^n$, but the reader should keep in mind that it is always possible to go back to the time series model.

C. Main Results

This part of our paper establishes necessary and sufficient conditions for the decoder (3) to be able to perfectly recover the state of each sensor. We define the set $\mathcal{C}(i) := \{j \in \mathcal{I} \mid c(j) = c(i)\}$, the set $A(i,n) := \{j \in \mathcal{I} - \{i\} \mid j \in \mathcal{I}_1(n) \text{ and } c(j) = c(i)\}$ and the set $B(i,n) := \{j \in \mathcal{I} - \{i\} \mid j \in \mathcal{I}_2(n) \text{ and } c(j) = c(i)\}$. We assume the following:

Assumption A

1) If $i \in \mathcal{I}_1(n)$, then $(\mid A(i,n) \mid)^{-1} \sum_{j \in A(i,n)} w_{ij}^n \leq \Phi_1$.
2) If $i \in \mathcal{I}_2(n)$, then $\Phi_2 \geq (\mid A(i,n) \mid)^{-1} \sum_{j \in A(i,n)} w_{ij}^n \geq \Phi_3$.
3) For every $(i,j) \in (\mathcal{I}_1(n))^2$, $w_{ij}^n \in [\underline{c}_1, \overline{c}_1]$, for every $(i,j) \in \mathcal{I}_1(n) \times \mathcal{I}_2(n)$, $w_{ij}^n \in [\underline{c}_2, \overline{c}_2]$ and for every $(i,j) \in (\mathcal{I}_2(n))^2$, $w_{ij}^n \in [\underline{c}_3, \overline{c}_3]$. Moreover $\underline{c}_1 \leq \underline{c}_2 \leq \overline{c}_1 \leq \overline{c}_2$, $\underline{c}_1 \leq \underline{c}_3 \leq \overline{c}_1 \leq \overline{c}_3$.

It is important to observe that Assumption A is only used in the analysis of the decoder, it is not needed for the implementation of any algorithm solving (3).

Theorem 1. ***Necessary conditions:*** *If assumption A is satisfied, and if*

$$\begin{cases} (1-\alpha)(\Phi_1 - \lambda) \mid A(i,n_1^*) \mid + \alpha C_{M_E}(\max\{\overline{c}_2, \overline{c}_3\} - \lambda) \mid \mathcal{C}(i) \mid + \alpha C_{M_E}(\overline{c}_2 - \lambda) \sum_{j \neq i} \mid \mathcal{C}(j) \mid \leq 0, \\ \\ (1-\alpha)(\Phi_3 - \lambda) \mid A(i,n_1^*) \mid + \alpha C_{M_E}(\underline{c}_2 - \lambda) \mid \mathcal{C}(i) \mid + \alpha C_{M_E}(\underline{c}_2 - \lambda) \sum_{j \neq i} \mid \mathcal{C}(j) \mid \geq 0, \end{cases} \tag{5}$$

with $n_1^* = \text{argmax}_n A(i,n)$*, then* m_i^* *is a local optimum of* (4)*.*

Sufficient conditions: *If assumption A is satisfied, if* (5) *is also satisfied and finally if*

$$\begin{cases} (1-\alpha)\left((\Phi_1-\lambda) \mid A(i,n_2^*) \mid +(\overline{c}_2-\lambda) \mid B(i,n_2^*) \mid\right) \\ +\alpha C_{M_E}(\max\{\overline{c}_2,\overline{c}_3\}-\lambda) \mid \mathcal{C}(i) \mid +\alpha C_{M_E}(\max\{\overline{c}_2,\overline{c}_3\}-\lambda) \sum_{j\neq i} \mid \mathcal{C}(j) \mid \leq 0, \\ \\ (1-\alpha)\left((\Phi_2-\lambda) \mid A(i,n_3^*) \mid +(\overline{c}_3-\lambda) \mid B(i,n_3^*) \mid\right) \\ +\alpha C_{M_E}(\max\{\overline{c}_2,\overline{c}_3\}-\lambda) \mid \mathcal{C}(i) \mid +\alpha C_{M_E}(\max\{\overline{c}_2,\overline{c}_3\}-\lambda) \sum_{j\neq i} \mid \mathcal{C}(j) \mid \leq 0, \end{cases} \tag{6}$$

where $n_2^* = \text{argmax}_n(\Phi_1-\lambda) \mid A(i,n) \mid +(\overline{c}_2-\lambda) \mid B(i,n) \mid$ *and* $n_3^* = \text{argmax}_n(\Phi_2-\lambda) \mid A(i,n) \mid +(\overline{c}_3-\lambda) \mid B(i,n) \mid$*, then* m_i^* *is the unique global optimum of* (3)*.*

Proof. See appendix. □

IV. Algorithms

In this section, we introduce two algorithms. The first one finds the global solution of (3), however, it is not useful if $I > 10$. The second algorithm finds a local solution of (3), but is much faster.

A. Global solution: Dynamic Programming

The decoder (3) is a nonlinear integer programming problem and it is interesting to note that this problem can be recast as a finite horizon Markov decision problem as stated in the next theorem.

Theorem 2. *Let* $p \in \{0,1\}^I$ *and* $n \in \{1,\ldots,N\}$*. The solution of the decoder* (3) *is also the solution of the following dynamic programming equation,* $\forall n \in \{1,\ldots,N\}$*:*

$$\begin{cases} V(p,n,1_p<1) &= \min_{u_i\in\{0,1\},\,\forall i} \sum_{(i,j)\in\mathcal{I}^2} e'_{ij} \\ \\ & \quad +V(up,n+1),\ \forall p \in \{0,1\}^I, \end{cases} \tag{7}$$

with the final condition $V(p,N+1)=0$ *for all* $p \in \{0,1\}^I$*, with* $e'_{ij} = e_{ij}(w_{ij}-\lambda)u_iu_jp_ip_j$ *and starting from the state* $(\mathbf{1},1)$*.*

Proof. We first introduce the new variable $u_i^n \in \{0,1\}$ for all $i \in \mathcal{I}$ and $n \in \{1,\ldots,N\}$. Then $m_i = \sum_{n=0}^N p_i^n$ with:

$$\begin{aligned} p_i^n &= u_i^n p_i^n,\ \forall i \in \mathcal{I},\ \forall n \in \{1,\ldots,N\}, \\ p_i^0 &= 1,\ \forall i \in \mathcal{I}. \end{aligned}$$

Therefore we have a deterministic system where the state is denoted by p_i^n and the control given by u_i^n. By simply looking at the cost function of (3), the statement of the theorem follows. □

The previous theorem seems to indicate that we could use all the reinforcement learning techniques to solve the drift detection problem. However, it is not feasible in a real-world setting because the number of states in the dynamic programming equation is equal to $N \times 2^I$, and most of the time, we will face situations where $I > 10$. Therefore, in the next section, we will derive a new algorithm, to obtain an approximate solution of (3).

B. Local solution: Ordinal Potential game

Knowing the complexity of solving the dynamic programming equation (7), we provide an algorithm which converges to a local optimum of (3). When the sufficient conditions of theorem 1 are satisfied, the proposed algorithm will converge to the global optimum. Our algorithm is based on a game theory reformulation of (3). The intuition is described below. We assume that every sensor/agent i is minimizing the following local function:

$$U_i(m_i, m_{-i}) = \sum_{n=1}^{m_i} \sum_{j \neq i}^{I} e_{ij} 1_{m_j^* > 0}(w_{ij}^n - \lambda).$$

A Nash equilibrium of this game is defined as the vector $\mathbf{m}^{NE} := [m_i^{NE}]_{1 \leq i \leq I}$ where for every i, m_i^{NE} satisfies:

$$U_i(m_i^{NE}, \mathbf{m}_{-i}^{NE}) \leq U_i(m, \mathbf{m}_{-i}^{NE}),\ \forall m \neq m_i^{NE},$$

where $\mathbf{m}_{-i} = [m_j]_{j \in \mathcal{I} - \{i\}}$. We are now able to prove that the previous optimization problem (3) can be seen as an ordinal potential (see [5] for a better understanding of potential games) of our game:

$$U_i(m_i, \mathbf{m}_{-i}) > U_i(m_i', \mathbf{m}_{-i}) \Leftrightarrow C(m_i, \mathbf{m}_{-i}) > C(m_i', \mathbf{m}_{-i}).$$

Therefore two observations can be made: (1) Each Nash Equilibrium is a local optimum of (3); (2) One-sided better reply dynamic will converge to a Nash equilibrium. The one-sided better reply dynamic is the following:

One-sided better reply algorithm

1) Set T.
2) For $t = 1, \ldots, T$ do:

a) For $i = 1, \ldots, I$ do:

$$m_i(t) = \text{argmin}_{m_i \in \{1,\ldots,N\}} U_i(m_i, m_{-i}(t)),$$

b) Set $\mathbf{m}(t+1) = \mathbf{m}(t)$.

3) Return $\mathbf{m}(T)$.

We do not provide a proof for the number of iterations needed for the convergence of the one-sided better reply dynamic, but we observe that in all our simulations (see section V), the convergence of the algorithm happens in less than 1s.

V. Numerical studies

This section is dedicated to the numerical studies performed to understand the robustness of our approach. We will first describe our two setups. Then we will discuss the different results obtained. Note that, in both scenario we assume that $M_{Eij} \sim \mathcal{N}(0, 0.01)$ and $M_{Eij} = M_{Eji}$, for all i, j. As a rule of thumb, we choose λ to be equal to the sample quantile, corresponding to the probability 0.7, of the vector $[w_{ij}^n]_{i,j,n}$. Through different simulations, this approximation has been proved to be efficient. In subsequent work, we will discuss what are the different options for λ. In all the numerical studies, we have only used the one-sided better reply algorithm.

A. Description of the simulation setups

Setup (1): In the first setup, we assume that $\mathcal{C} := \{1, 2\}$ and $s_1^n \sim \mathcal{N}(0, 0.1)$ and $s_2^n \sim \mathcal{N}(1.2, 0.1)$ for every n. Moreover $c(i) = 1$ for all $i \in \{1, \ldots, \lfloor I/2 \rfloor\}$ and $c(i) = 2$ otherwise. Concerning the drift, sensors are picked randomly as well as the instant at which a sensor starts to drift. Finally if the sensor i drifts, then $x_i^n \sim \mathcal{N}(\mu_i, 0.1)$ for every n. In this setup, we choose $\mu_i = (2x_i - 1)(i + 1)$, where $x_i \sim \text{Bernouilli}(0.5)$. The major purpose of this setup is to prove and to ensure the performance of our algorithm. It does not have any ambition of capturing a real-world scenario. An illustration of this setup, with four drifted sensors, is presented in figure 1.

Setup (2): In the second setup, we will try to capture a more realistic scenario. Here, we assume that $\mathcal{C} := \{1, 2\}$, the source signals are ARMA processes for which the parameters are also

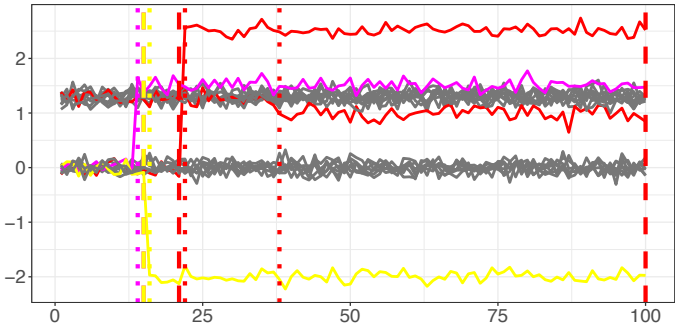

Figure 1. Setup (1): The measurements which did not drift are in grey in this figure. The drift measurements are in red, purple and yellow. The vertical dotted lines represented the instant at which sensors drifted and the vertical dashed lines capture the instant predicted by the decoder.

chosen randomly. As in [7], we assume that the classical low-cost sensor drift model is similar to the increase in the variance due to the gradual acquisition of drift. We introduce degradation at a randomly selected time n for the sensor i as $\delta_i^n = \delta_i^n + \epsilon_i^n$, where $\epsilon_i^n \sim \mathcal{N}(0.01, 0.01)$. Again, the drift sensors are picked randomly. This setup with four sensors, is illustrated in figure 2.

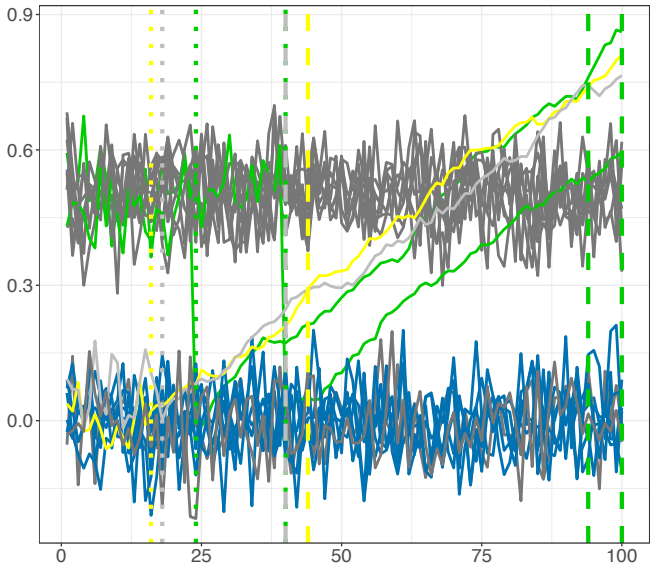

Figure 2. Setup (2): The measurements which did not drift are in grey and dark blue in this figure. The measurements in dark blue indicate that the decoder predicted, wrongly, that these measurements are unreliable. The drift measurements are in green, light grey and yellow. As in figure 1, the vertical dotted lines represented the instant at which sensors drifted and the vertical dashed lines capture the instant of drift beginning predicted by the decoder.

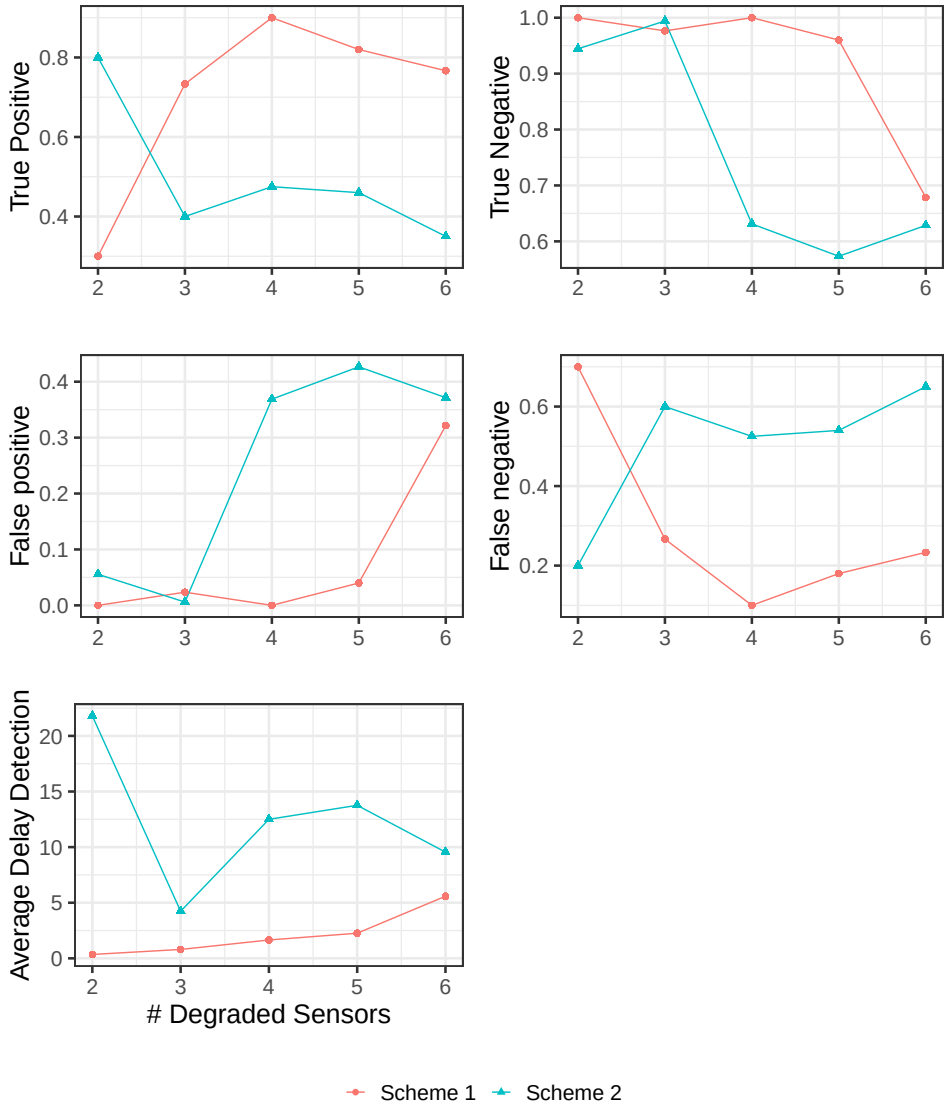

Figure 3. Performance metrics.

B. Results

We define two performance metrics: The *average detection delay* and the classical rate of *False Positive/Negative*. If a sensors are designed to fail in a predetermined number of trials, the average delay is the mean of the absolute difference between the true instant of drift acquisition and the estimated time of degradation. Fig.3 represents the results of the One-sided better reply algorithm. For each point on the x-axis, the detection time and the rate of *False Positive/Negative* are averaged over 10 trials. As expected, the decoder is performing better in setup 1 than setup 2.

In setup 1, we observe an unexpected phenomenon. The rate of false-negative decreases when the number of unreliable sensors increases. Otherwise, the decoder behaves as expected in the setup, its performance decreases when the number of unreliable sensors increases. Again, in setup 2, when the number of failed sensors increases, the performance of the decoder decreases in terms of the rate of false positive and false negative. However, when a drifted sensor is detected by the decoder, we observe that the average detection delay of detection is slightly better when the number of unreliable sensors increases.

APPENDIX

Proof. Step 1 (Necessary condition): We first prove the necessary condition for $\mathbf{m}^*$ to be a local optimum of (3). If $m' < m_i^*$, then from (4) we can decomposed $\sum_{n=m'+1}^{m_i^*} \sum_{j=1}^{I} e_{ij} 1_{m_j^* \geq n}(w_{ij}^n - \lambda)$ into the sum of two terms $\sum_{n=m'+1}^{m_i^*} \sum_{j \in c(i)}^{I} e_{ij} 1_{m_j^* \geq n}(w_{ij}^n - \lambda)$ and $+\sum_{n=m'+1}^{m_i^*} \sum_{j \neq c(i)}^{I} e_{ij} 1_{m_j^* \geq n}(w_{ij}^n - \lambda)$. From assumption A, note that for all $n \leq m_i^*$, $\sum_{j \in C(i)}^{I} 1_{m_j^* \geq n}(w_{ij}^n - \lambda) \leq A(i,n)(\Phi_1 - \lambda)$, we have

$$
\begin{aligned}
& \sum_{n=m'+1}^{m_i^*} \sum_{j \in C(i)}^{I} \alpha M_{ij} 1_{m_j^* \geq n}(w_{ij}^n - \lambda) \\
\leq \; & (m_i^* - m' - 1) \mid \mathcal{C}(i) \mid \alpha C_{M_E}(\bar{c}_3 - \lambda), \\
& \sum_{n=m'+1}^{m_i^*} \sum_{j \neq C(i)}^{I} \alpha M_{ij} 1_{m_j^* \geq n}(w_{ij}^n - \lambda) \\
\leq \; & (m_i^* - m' - 1) \sum_{j \neq i} \mid \mathcal{C}(j) \mid \alpha C_{M_E}(\bar{c}_3 - \lambda)
\end{aligned}
$$

Therefore it follows that if (5), then $\sum_{n=m'+1}^{m_i^*} \sum_{j=1}^{I} e_{ij} 1_{m_j^* \geq n}(w_{ij}^n - \lambda) \leq 0$. A similar line of thought can be used for every $m' > m_i^*$. Instead of find an upper bound we need to find a lower bound for $\sum_{n=m_i^*+1}^{m'} \sum_{j=1}^{I} e_{ij} 1_{m_j^* \geq n}(w_{ij}^n - \lambda)$.

Step 2 (Uniqueness condition): Let us assume that there exists another local solution $\mathbf{m}^{loc} \neq \mathbf{m}^*$. We will use a contradiction argument. We need to prove that it exists m', such that (4) is not satisfied for $\mathbf{m}^{loc}$. Two cases need to be solved. First, for a given i, let us assume that

$m_i^{loc} < m' \leq m_i^*$. Note that $i \in \mathcal{I}_1(m')$ and from (4) we have:

$$\begin{aligned}
&\sum_{n=m_i^{loc}+1}^{m'} \sum_{j=1}^{I} e_{ij} 1_{m_j^{loc} \geq n} (w_{ij}^n - \lambda) \\
= &\sum_{n=m_i^{loc}+1}^{m'} \sum_{j \in C(i)}^{I} ((1-\alpha) + \alpha M_{ij}) 1_{m_j^{loc} \geq n} (w_{ij}^n - \lambda) \\
&+ \sum_{n=m_i^{loc}+1}^{m'} \sum_{j \neq C(i)}^{I} \alpha M_{ij} 1_{m_j^{loc} \geq n} (w_{ij}^n - \lambda)
\end{aligned}$$

From assumption A, from $B(i,n) \leq \overline{B}_{c(i)}$ and from $\mid A(i,n) \mid = \mid \mathcal{C}(i) \mid - \mid B(i,n) \mid$ we can easily deduce the following inequality:

$$\begin{aligned}
&\sum_{n=m_i^{loc}+1}^{m'} \sum_{j \in C(i)}^{I} 1_{m_j^{loc} \geq n} (w_{ij}^n - \lambda) \\
\leq &\sum_{n=m_i^{loc}+1}^{m'} (\Phi_1 - \lambda) \mid A(i,n) \mid + (\overline{c}_2 - \lambda) \mid B(i,n) \mid \\
\leq &\ (1-\alpha)(m' - m_i^{loc} - 1) \\
&\times ((\Phi_1 - \lambda) \mid A(i,n^*) \mid + (\overline{c}_2 - \lambda) \mid B(i,n^*)) \mid,
\end{aligned}$$

where $n^* = \text{argmax}_n (\Phi_1 - \lambda) \mid A(i,n) \mid + (\overline{c}_2 - \lambda) \mid B(i,n) \mid$. Moreover, by using assumption A, we have:

$$\begin{aligned}
&\sum_{n=m_i^{loc}+1}^{m'} \sum_{j \in C(i)}^{I} \alpha M_{ij} 1_{m_j^{loc} \geq n} (w_{ij}^n - \lambda) \\
\leq &\ (m' - m_i^{loc} - 1) \mid \mathcal{C}(i) \mid \alpha C_{M_E} (\overline{c}_3 - \lambda) \\
&\sum_{n=m_i^{loc}+1}^{m'} \sum_{j \neq C(i)}^{I} \alpha M_{ij} 1_{m_j^{loc} \geq n} (w_{ij}^n - \lambda) \\
\leq &\ (m' - m_i^{loc} - 1) \sum_{j \neq i} \mid \mathcal{C}(j) \mid \alpha C_{M_E} (\overline{c}_3 - \lambda)
\end{aligned}$$

Note that if the assumption of the theorem are respected then it is leading us to a contraction. Indeed, $\sum_{n=m_i^{loc}+1}^{m'} \sum_{j=1}^{I} e_{ij} 1_{m_j^{loc} \geq n} (w_{ij}^n - \lambda) < 0$ and not be positive.

The second case is the following, for a given i, let us assume that $m_i^* < m' \leq m_i^{loc}$. Note that $i \in \mathcal{I}_2(m')$. Then by using the same argument as previously and by replacing $(\Phi_1 - \lambda) \mid A(i,n) \mid + (\overline{c}_2 - \lambda) \mid B(i,n) \mid$ by $(\Phi_2 - \lambda) \mid A(i,n) \mid + (\overline{c}_3 - \lambda) \mid B(i,n) \mid$, we will reach the same conclusion.

□

REFERENCES

[1] Tamer Başar and Pierre Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[2] Toby Dylan Hocking, Armand Joulin, Francis Bach, and Jean-Philippe Vert. Clusterpath an algorithm for clustering using convex fusion penalties. In *28th international conference on machine learning*, page 1, 2011.

[3] Itamar Katz and Koby Crammer. Outlier-robust convex segmentation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[4] Dheeraj Kumar, Sutharshan Rajasegarar, and Marimuthu Palaniswami. Automatic sensor drift detection and correction using spatial kriging and kalman filtering. In *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 183–190. IEEE, 2013.

[5] Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[6] MS Takruri, Subhash Challa, and Rajib Chakravorty. Recursive bayesian approaches for auto calibration in drift aware wireless sensor networks. *Journal of Networks*, 2010.

[7] Yuzhi Wang, Anqi Yang, Zhan Li, Xiaoming Chen, Pengjun Wang, and Huazhong Yang. Blind drift calibration of sensor networks using sparse bayesian learning. *IEEE Sensors Journal*, 16(16):6249–6260, 2016.