sensors

MDPI

Article

Application-Aware Anomaly Detection of Sensor Measurements in Cyber-Physical Systems

Amin Ghafouri [1], Aron Laszka [2], and Xenofon Koutsoukos [1,*]

[1] Department of Electrical Engineering and Computer Science, Institute for Software Integrated Systems (ISIS), Vanderbilt University, Nashville, TN 37212, USA ; aminghafouri.ut@gmail.com
[2] Department of Computer Science, University of Houston, Houston, TX 77204, USA; alaszka@uh.edu
* Correspondence: Xenofon.Koutsoukos@vanderbilt.edu; Tel.: +1-615-322-8283

Received: 24 May 2018; Accepted: 20 July 2018; Published: 27 July 2018

Abstract: Detection errors such as false alarms and undetected faults are inevitable in any practical anomaly detection system. These errors can create potentially significant problems in the underlying application. In particular, false alarms can result in performing unnecessary recovery actions while missed detections can result in failing to perform recovery which can lead to severe consequences. In this paper, we present an approach for *application-aware anomaly detection (AAAD)*. Our approach takes an existing anomaly detector and configures it to minimize the impact of detection errors. The configuration of the detectors is chosen so that application performance in the presence of detection errors is as close as possible to the performance that could have been obtained if there were no detection errors. We evaluate our result using a case study of real-time control of traffic signals, and show that the approach outperforms significantly several baseline detectors.

Keywords: anomaly detection; detection error; cyber-physical systems; traffic sensors

1. Introduction

Sensors deployed in cyber-physical systems (CPS) for monitoring and control purposes are prone to anomalies (e.g., reliability failures and cyber-attacks). To detect anomalies and prevent their harmful effects, anomaly detection systems (ADS) are utilized. However, ADS suffer from false positives (i.e., false alarms) and false negatives (i.e., missed detections), which may result in high performance degradation in CPS applications. In particular, false positives result in recovery that is not required, and false negatives result in failing to perform recovery when it is indeed required. Such detection errors can cause incorrect measurements being transmitted to a controller, and thus result in obtaining non-optimal or even destabilizing control decisions, which may compromise the performance of the system. For example, detection errors may result in disastrous events, such as reactor explosion in process control systems, water contamination in water distribution networks, and extremely heavy traffic congestion in intelligent transportation systems [1,2].

To address the challenges caused by detection errors, it is necessary to take into account the CPS application when designing anomaly detectors, and to quantify the losses in the application caused by potential detection errors. To minimize the losses, it is desirable to reduce the detection errors as much as possible. However, there exists a trade-off between them (i.e., decreasing the rate of false alarms may increase the rate of missed faults, and vice versa), which can be changed through a detection threshold. Therefore, the performance loss caused by detection errors can be minimized by selecting the right detection threshold.

Our goal is to perform these steps using a novel approach, which takes an existing anomaly detector and configures it considering the behavior of the controller. We call our framework *Application-Aware Anomaly Detection (AAAD)*. This framework takes into account the interactions

between the controller and the application, and so it can compute how each detection decision may affect the underlying application. Knowing this, the detector attempts to make detection decisions that will result in the least performance loss in the underlying application if the detection decision is not accurate due to false positive and false negative errors.

Previous works have proposed different anomaly detection methods for CPS [1]. In addition, there is a wide body of literature on machine learning-based anomaly detection [3]. The problem of finding optimal thresholds for intrusion detectors is studied in [4]. The paper shows that computing optimal attacks and defenses is computationally expensive, and proposes heuristic algorithms for computing near-optimal strategies. Further, the work in [5] studies the problem of finding optimal thresholds for anomaly-based detectors implemented in dynamical systems in the face of strategic attacks. The paper provides algorithms to compute optimal thresholds that minimize losses considering best-response attacks. However, there is little work that takes into account the tight interaction between the detector and the controller of a CPS, which as we show in this work if taken into account, can result in improved performance and robustness.

Contributions

In this paper, we propose an application-aware anomaly detection framework for minimizing the impact of detection errors in CPS. The contributions of AAAD are as follows:

1. We devise an effective detector for identifying anomalies in sensor measurements using machine learning regression and an approach to recover from anomalies in order to maintain operation when detection alerts are triggered.
2. We formulate the AAAD problem, in which a detector is optimally configured such that the performance loss in the presence of detection errors is minimized. In particular, the thresholds are selected so that the performance of the system in the presence of detection errors is as close as possible to the performance that could have been achieved if there were no detection errors.
3. We show that the AAAD problem is computationally challenging, and then we present an efficient algorithm to find near-optimal solutions.
4. We analyze two special cases of the application-aware detection problem: (a) single detector and (b) detectors with equal thresholds. We present optimal solutions for both special cases which can provide insights into the novelty of the approach.
5. We evaluate AAAD using simulation experiments on a case study of real-time control of traffic signals. The evaluation results demonstrate the benefits of the approach compared with standard anomaly detection techniques.

We believe that the proposed approach can be useful in any system where there are a significant number of sensors with high variations in sensor values, which may cause many false-positive and false-negative errors. A real-world example of such a CPS application would be real-time control of traffic signals since in large cities, there are thousands of sensors that could become anomalous. Throughout the paper, we will use traffic sensors as a running example to illustrate the application of our framework, but we present the model and results using a general, domain-agnostic formulation. Alternative examples of possible application domains include sensors in water-distribution networks [6].

Related Work

Anomaly detection in cyber-physical systems presents an important and challenging problem [7,8]. As a result, a variety of approaches have been proposed in prior work on anomaly detection. In contrast to our approach, these prior results focused on the design of detectors rather than the optimal configuration of an existing detector.

Several detectors have been built on machine learning and, in particular, neural networks. Goh et al. present an unsupervised approach for anomaly detection in cyber-physical systems based on

recurrent neural networks and the cumulative sum method [9]. Kosek presents a contextual anomaly detection method for smart grids based on neural networks [10]. Krishnamurthy et al. use an alternative model, Bayesian networks [11]. They present an approach for learning causal relations and temporal correlations in cyber and physical variables from unlabeled data using Bayesian networks. These networks can then be used to detect anomalies and isolate their root causes.

Jones et al. propose a formal-methods-based approach for anomaly detection in cyber-physical systems [12]. They introduce a model-free, unsupervised learning procedure that constructs a signal temporal logic (STL) formula from system output data collected during normal operations. Then, anomalies can be detected by flagging system trajectories that do not satisfy the learned formula. In a follow-up work, Kong et al. describe a formal-methods-based approach for supervised anomaly learning [13]. Chibani et al. investigate the problem of designing fault-detection filters for fuzzy systems, considering faults and unknown disturbances in discrete-time polynomial fuzzy systems [14,15]. A diagnostic observer-based system for fault detection of fuzzy systems which optimizes the worst case robustness and fault sensitivity is presented in [16].

Anomaly detection has also been considered in the context of security intrusions, that is, to detect cyber-physical attack against a CPS [17]. For example, Urbina et al. study physics-based detection of stealthy attacks against industrial control systems [1]. They review prior work on attack detection and argue that many of these use detection schemes that do not limit the impact of stealthy attacks. They then propose a new metric for measuring impact and demonstrate that attacks may be detected with a proper configuration. In contrast, Kleinmann and Wool consider detection of attacks against industrial control systems based on cyber anomalies [18].

Besides anomaly detection, other approaches have also been considered for detecting faults or attacks in traffic networks. Lu et al. review prior work on the problem of anomaly detection of traffic sensors [19]. Based on the level of data used, they divide detection methods into three levels: macroscopic (highly-aggregated data), mesoscopic (synchronized data for a section of freeways), and microscopic. They also review data-correction methods and provide practical guidelines for anomaly detection in traffic applications. Zygouras et al. present three methods—based on Pearson's correlation, cross-correlation, and multivariate ARIMA—to detect faulty traffic measurements [20]. They discuss the performance of all three methods and demonstrate that they are complementary to each other. Further, they employ crowd-sourcing to resolve whether irregular measurements are due to faulty sensors or unusual traffic. Finally, Robinson presents a test, based on the relationship between flows at adjacent sensors, to detect faulty loop detectors [21].

Nonetheless, to the best of our knowledge, none of these papers consider the performance of the controller in the design and configuration of anomaly detectors, and they do not perform any application-aware optimization to improve detection performance.

Paper Outline

The rest of this paper is organized as follows. In Section 2, we introduce the system model. In Section 3, we discuss the regression-based anomaly detection. In Section 4, we present the application-aware detection problem for detection error-tolerant selection of thresholds in anomaly detectors. In Section 5, we analyze the application-aware detection problem and present an algorithm to obtain near-optimal solutions. In Section 6, we study two special variations of the application-aware detection problem, that is, single detector and detectors with equal threshold. In Section 7, we evaluate our approach numerically using a case study of real-time control of traffic signals. Finally, we offer concluding remarks in Section 8.

2. Model

In this section, we present the system model and informally introduce the problem of application aware anomaly detection. We also present a running example of real-time control of traffic signals that is used throughout the paper to demonstrate our approach.

2.1. Notation

Vectors are denoted by bold symbols. Vector $\boldsymbol{y}$ at timestep k is described by $\boldsymbol{y}^k$. We omit the timestep symbol when all symbols have same timestep k. However, timestep symbol is used when there are different timesteps present or when it eases understanding. Given vector $\boldsymbol{y}$ and set of indices I, vector $\boldsymbol{y}_I$ is defined as a vector with same size as $\boldsymbol{y}$ that has the same size as $\boldsymbol{y}$ for indices in I, and is zero otherwise. For a list of symbols used in this section, see Table 1.

Table 1. List of Symbols.

Symbol	Description
S	Set of sensors
a_s	Actual value for sensor s
m_s	Measured value for sensor s
p_s	Predicted value for sensor s
$TP_s(\tau)$	True positive probability of the detector for sensor s given detection threshold τ
$FP_s(\tau)$	False positive probability of the detector for sensor s given detection threshold τ
$TN_s(\tau)$	True negative probability of the detector for sensor s given detection threshold τ
$FN_s(\tau)$	False negative probability of the detector for sensor s given detection threshold τ
w_s	Recovered measurement transmitted to the controller for sensor s
r_s	Residual signal for sensor s

2.2. System Model

Consider a CPS, for example, an intelligent transportation system or a process control system, consisting of a plant and a controller as shown in Figure 1. At each timestep, given measurements $\boldsymbol{w}$ containing information about the system, a controller computes a control input u that maximizes a utility function $J(\boldsymbol{w}, u)$. In other words, the controller determines the optimal control input u^* defined as

$$u^* \in \underset{u}{\operatorname{argmax}}\, J(\boldsymbol{w}, u)\ , \tag{1}$$

where the optimal utility is denoted by $J^*(\boldsymbol{w})$.

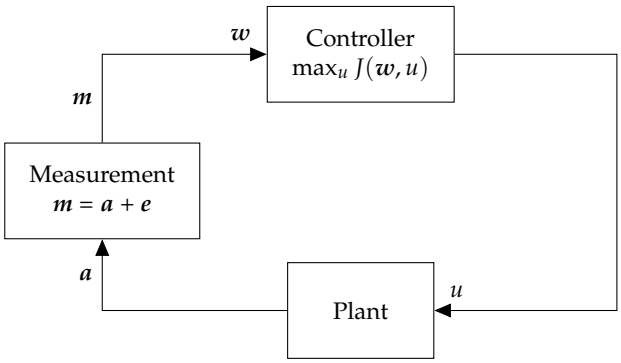

Figure 1. System Model. Please note that in this case $\boldsymbol{w} = \boldsymbol{m}$.

Anomalous Sensors

Sensors may be anomalous due to hardware failures or sensor attacks. If sensor $s \in S$ is anomalous, there is a discrepancy between the actual and observed measured values. In other words, if a_s is the actual value and m_s is the observed measurement at a timestep, for an anomalous sensor we have $m_s = a_s + e_s$, where $e_s \in \mathbb{R}$ is the error value at that timestep.

2.3. Example: Real-Time Traffic Signal Control

We present real-time control of traffic signals as a running example that is used throughout the paper to demonstrate the applicability of our approach. In the following, we describe the widely-popular max-pressure controller for optimal control of traffic signals [22]. In the original max-pressure algorithm presented in [22], the traffic state is represented using exogenous demands that are then routed through the network using routing ratios. In this work, instead of using exogenous demands that are then transformed to internal demands through using routing ratios, we assume that the internal demands are directly provided. Please note that this does not affect the max-pressure algorithm as the algorithm effectively uses internal demands in its computations.

Max-Pressure Controller

Consider a network of intersections I with road links L. Movement from a link $i \in L$ to a link $j \in L$ is denoted by a pair $(i,j) \in E$. Further, let each movement (i,j) have a queue associated with it, and at each timestep, let $x(i,j)$ represent the length of this queue. The length of the queue shows how many vehicles intend to travel from i to j. For each movement (i,j), the pressure is defined as

$$P(i,j) = x(i,j) - \sum_{p} x(j,p) \ ,$$

which is simply the number of cars in the queue minus the total number of cars in the downstream queues.

Each intersection n has a traffic signal with a set of admissible stages Φ_n. Each stage $u_n \in \Phi_n$ is a set of simultaneous movements that are permitted by the traffic signal. If u_n permits a movement (i,j), then $u_n(i,j) = 1$, otherwise $u_n(i,j) = 0$. Let $c(i,j)$ be the saturation flow of movement (i,j). Given a stage $u_n \in \Phi_n$, pressure-release (i.e., utility) for intersection n is defined as

$$J_n(u_n) = \sum_{i,j} c(i,j)\, P(i,j)\, u_n(i,j) \ .$$

Traffic sensors can be used to measure $x(i,j)$ for all $(i,j) \in E$ and they are prone to failures.

Algorithm 1 presents the max-pressure (MP) controller in detail. At each intersection n, the MP controller selects the stage u_n that results in the maximum pressure-release. In other words, the MP controller computes

$$u_n^* \in \underset{u_n \in \Phi_n}{\operatorname{argmax}} \sum_{i,j} c(i,j)\, P(i,j)\, u_n(i,j) \ . \tag{2}$$

Please note that at each intersection, the MP control selects a stage that depends only on the queues adjacent to the intersection. It is shown that the MP controller maximizes network throughput [22].

Algorithm 1 Max-Pressure Controller [22].

Input: $x(i,j)$ for all $(i,j) \in E$
for all $n \in I$ **do**
  for all $(i,j) \in E$ **do**
    $P(i,j) \leftarrow x(i,j) - \sum_p x(j,p)$
  end for
  $u_n^* \leftarrow \operatorname{argmax}_{u_n \in \Phi_n} \sum_{i,j} c(i,j)\, P(i,j)\, u_n(i,j)$
end for
return $\{u_n^*\}_{n \in I}$

The overall utility for traffic network can be calculated by adding individual utilities for the intersections. That is, $J(x,u) = \sum_n J_n(x,u_n)$ where $u = \{u_n\}_{n \in I}$. Please note that using this representation, the MP optimization problem becomes the same as (1).

2.4. Anomaly Detection, Recovery, and Resilience

Next, we discuss anomalies caused by sensor faults and their detection, and we informally introduce the problem of optimal detector configuration, which we will formalize in Section 4.4. In contrast to the previous subsection, where we introduced our running example, the discussion here will again be domain agnostic.

Anomalies may cause damage to the system and significantly degrade the performance of the CPS application (as measured by the utility function $J(\boldsymbol{w},u)$). Consequently, we must employ an anomaly detector to detect faults in sensor measurements. In our running example, anomalies can affect the traffic measurements, and therefore, degrade the utility of the traffic signal control. Suppose that we have an anomaly detection method. Upon detection, the system must recover from anomalies and continue operation. To this end, we must employ a recovery method in order to continue operation in the presence of detection alerts. In the traffic signal control example, operation must continue even in the presence of anomalous traffic measurements.

Now, suppose that we have a recovery approach that computes a recovered vector of measurements in the presence of detection alerts. In anomaly detectors, there are detection errors, that is, false positives (i.e., false alarms) and false negatives (i.e., missed faults). If there were no detection errors, the recovered vector of measurements would be close to the actual values (of course assuming that the recovery approach works well). However, in the presence of detection errors, false positives result in recovery that is not required, and false negatives result in failing to perform recovery when it is indeed needed.

To illustrate the effect of detection errors on the application, let $\boldsymbol{w}'$ denote the recovered measurement vector, which will result in the utility $J' = \max_u J(\boldsymbol{w}',u)$. However, if there were no detection error, we could have obtained the optimal utility $J^* = \max_u(\boldsymbol{a},u)$. Hence, we face the problem of optimally configuring an anomaly detector through the selection of detection thresholds so that the actual obtained utility in the presence of detection errors (i.e., J') is as close as possible to the utility that would have been obtained if there were no detection errors (i.e., J^*). In the traffic signal control example, the utility is quantified by the pressure release for an intersections that can be used as a proxy for maximizing network throughput.

3. Anomaly Detection

In this section, we construct an example regression-based anomaly detector for identifying anomalous sensor measurements. We then discuss detection errors and some metrics that are used to characterize them.

3.1. Regression-Based Anomaly Detector

To protect the system against anomalies, we must detect them quickly and accurately. Many different anomaly detection systems have been proposed in the literature. For a comprehensive review of anomaly detection methods, we refer the reader to [3] for machine-learning-based detectors and [1] for detectors used in CPS. In this work, we use regression-based anomaly detectors because in addition to high detection performance, such detectors require no knowledge of the physical system, can take into account complex and nonlinear behaviors of the system, and are easy to implement and can be highly scalable.

3.1.1. Architecture

Figure 2 shows the architecture of regression-based anomaly detector. The detector consists of two main components: (1) Predictor and (2) Statistical Test. The predictor predicts the value of a sensor

given some information about the system state (e.g., current value of other sensors, previous control inputs). Then, the statistical test compares the computed prediction to the observed measurement and decides whether the sensor is normal or anomalous. We describe each component in more detail considering the running example of real-time traffic control.

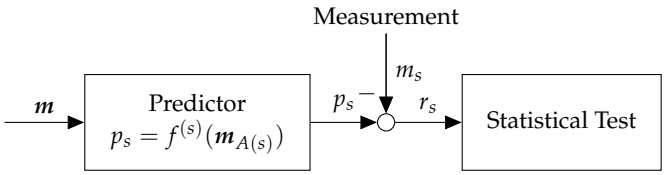

Figure 2. Regression-Based Anomaly Detector.

3.1.2. Predictor

Our goal is to find a function $f^{(s)}$ that maps spatial or temporal features to the actual value of a sensor s (e.g., traffic flow or occupancy). In practice, two traffic sensors are highly correlated if they are in close proximity. Thus, we let the features be the measured values of other adjacent sensors at the same timestep, denoted by $\boldsymbol{m}_{A(s)}$ where $A(s)$ is a set of sensors adjacent to $A(s)$ found using cross-validation. Please note that this approach is particularly applicable to traffic networks as there are usually many redundant sensors in the network. The function $f^{(s)}$ can then be obtained using suitable machine learning regression algorithm such as deep neural networks [23], Gaussian Processes [24], and many others [25]. Thus, for sensor s, we obtain the prediction as $p_s = f^{(s)}(\boldsymbol{m}_{A(s)})$.

3.1.3. Statistical Test

The statistical test efficiently detects anomalies for each sensor $s \in S$ by comparing the measured value $m_s(k)$ with the predicted value $p_s(k)$. Given a set of measured values $\boldsymbol{m} = \langle m_s \rangle_{s \in S}$ and predicted values $\boldsymbol{p} = \langle p_s \rangle_{s \in S}$, residual signals are computed as $\boldsymbol{r} = |\boldsymbol{m} - \boldsymbol{p}|$. Then, given the residuals, the statistical test makes detection decisions $\boldsymbol{d} = \langle d_s \rangle_{s \in S}$, where for each sensor s, the decision d_s is either normal or anomalous.

Different detection algorithms can be used to implement the statistical test [26]. In this work, we consider a stateless threshold-based detector defined as follows. It should be noted that for ease of presentation, our model assumes that a detector can be configured using a single threshold value τ_s. However, our results can be easily applied to detectors that are configured using multiple parameter values. We discuss how to incorporate such detectors into our framework in Section 3.2. Given detection thresholds $\boldsymbol{\tau} = \langle \tau_s \rangle_{s \in S}$, for each sensor s, if the residual r_s is less than or equal to the threshold τ_s, then s is marked normal and otherwise, s is marked anomalous. Thus

$$d_s = \begin{cases} \text{normal } (s \in N) & \text{if } r_s \leq \tau_s \\ \text{anomalous } (s \in A) & \text{otherwise} \end{cases}. \tag{3}$$

3.2. Detection Error

In anomaly detectors, there might be a *false negative*, which means failing to raise an alarm when an anomaly did happen. Further, there might be a *false positive*, which means raising an alarm when the system exhibits normal behavior. It is desirable to reduce the false positive and false negative probabilities as much as possible. However, there exists a trade-off between them, which can be controlled by changing the detection threshold. In particular, by decreasing (increasing) the threshold, one can decrease (increase) the FN probability and increase (decrease) the FP probability.

We represent the FN probability for each sensor s by the function $FN_s : \mathbb{R}_+ \rightarrow [0,1]$, where $FN_s(\tau_s)$ is the probability of FN when the threshold is τ_s, given that the sensor is anomalous. Similarly, we denote the attainable FP probability for each sensor s by $FP_s : \mathbb{R}_+ \rightarrow [0,1]$, where $FP_s(\tau_s)$ is

the FP probability when the threshold is τ_s, given that the sensor is in normal operation. The true positive and true negative probabilities are also denoted by $TP_s(\tau_s)$ and $TN_s(\tau_s)$. Clearly, we have $TP_s(\tau_s) = 1 - FN_s(\tau_s)$ and $TN_s(\tau_s) = 1 - FP_s(\tau_s)$.

It should be noted that even though we assumed in Section 3.1.3 that each detector can be configured using a single threshold value τ_s, our framework can actually be applied to detectors that are configured using multiple parameter values. For such a detector, each possible configuration (i.e., combination of parameter values) results in some pair of false-negative and false-positive error probabilities. By considering the Pareto optimal configurations (i.e., configurations such that neither probability can be decreased without increasing the other), we can obtain a curve that represents the best attainable pairs of false-negative and false-positive probabilities. Then, we can simply let threshold values correspond to points on this curve such that false-negative and false-positive probabilities (FN_s and FP_s) are increasing and decreasing functions of the threshold τ_s, respectively.

4. Application-Aware Anomaly Detection

In this section, we present the problem of application-aware anomaly detection (AAAD). First, we describe an approach for recovery in order to continue operation in the presence of detection alerts. Then, we quantify the utility losses in the application caused by potential detection errors. Based on the characterization of the utility losses, we formulate the AAAD problem, i.e., the problem of finding detection thresholds so that the obtained utility in the presence of detection errors is as close as possible to the utility that could have been obtained if there were no detection errors.

4.1. Architecture

Figure 3 shows the AAAD architecture. If there is a detection alert, the prediction is routed to the application, instead of the measurement. The threshold of each detector is selected such that in the presence of detection error, the routed value (i.e., measurement or prediction) still obtains a utility close to the utility that could have been obtained if there were no detectors. (Please note that in the figure, the predictor is not connected to the anomaly detector since this framework is applicable to any threshold-based detector, and not only regression-based detectors.)

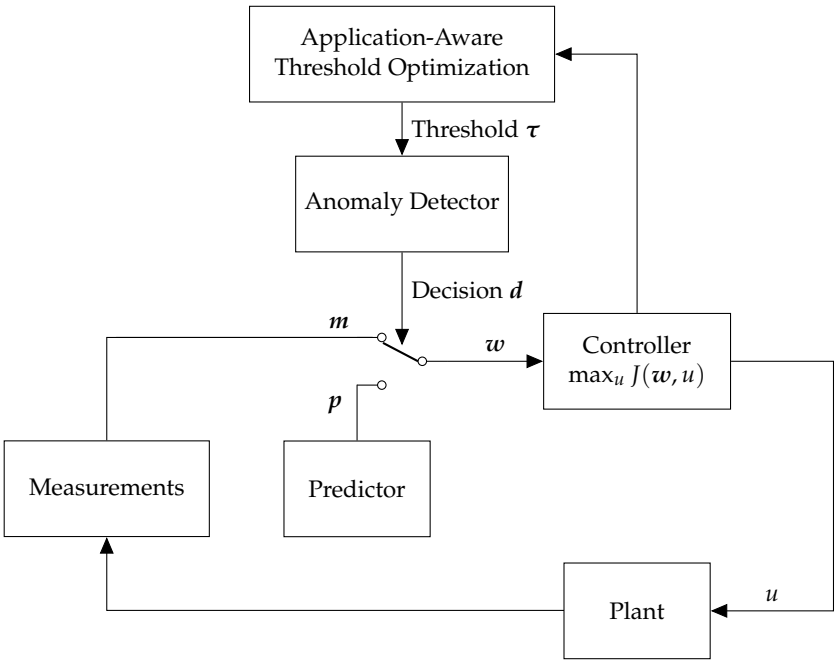

Figure 3. Architecture of Application-Aware Anomaly Detection.

4.2. Recovery

We consider a recovery approach in order to continue operation in the presence of detection alerts. If sensor s is marked normal, then the observed measurement m_s is transmitted to the controller. However, if sensor s is marked anomalous, then the observed measurement is discarded and instead, the prediction p_s is transmitted to the controller. The switch in Figure 3 illustrates this idea. To formally represent the recovery approach, let w_s denote the *recovered measurement* transmitted to the controller. Then, w_s can be described as

$$w_s = \begin{cases} m_s & \text{if } s \text{ is normal} \\ p_s & \text{if } s \text{ is anomalous} \end{cases}. \tag{4}$$

For the threshold-based detector defined by (3), the measurement of sensor s is marked normal if $|p_s - m_s| \leq \tau_s$ and anomalous otherwise. Therefore, for threshold-based detectors, the above equation can be re-written as

$$w_s(\tau_s) = \begin{cases} m_s & \text{if } |m_s - p_s| \leq \tau_s \\ p_s & \text{otherwise} \end{cases}. \tag{5}$$

Please note that in this case, given prediction p_s and measurement m_s, the value of w_s depends on the threshold τ_s. To highlight this dependence, we use the notation $w_s(\tau_s)$ instead of w_s. To summarize, given vectors of predictions $\boldsymbol{p}$, measurements $\boldsymbol{m}$, and thresholds $\boldsymbol{\tau}$, using (5), we are able to compute the recovered measurement vector $\boldsymbol{w}(\boldsymbol{\tau})$ that is transmitted to the controller.

We assume that when a measurement is normal, it provides the best obtainable value for the sensor. Also, we assume that when a measurement is anomalous, the prediction provides the best obtainable value for the sensor.

4.3. Worst-Case Utility Loss Due to Detection Error

The control input u (i.e., defined by (1)) depends on the recovered measurements $\boldsymbol{w}(\boldsymbol{\tau})$ (i.e., defined by (5)), and the recovered measurements $\boldsymbol{w}(\boldsymbol{\tau})$ depend on the detection thresholds $\boldsymbol{\tau}$. Therefore, the value of control input depends on thresholds $\boldsymbol{\tau}$. For example, if the thresholds are small (large), there will be many (few) detection alarms, and so predictions (measurements) will often be transmitted to the controller. Unfortunately, this will be problematic in the presence of detection errors.

Given threshold $\boldsymbol{\tau}$, let N be the set of sensors that are marked normal (i.e., $\forall s \in N, r_s \leq \tau_s$) and let A be the set of sensors that are marked anomalous (i.e., $\forall s \in A, r_s > \tau_s$). Based on the recovery method (5), the predictions are used for marked-anomalous sensors in A and measurements are used for marked-normal sensors in N to create the recovered measurement vector, i.e., $\boldsymbol{w} = \boldsymbol{p}_A \cup \boldsymbol{m}_N$. Next, given the recovered measurements $\boldsymbol{p}_A \cup \boldsymbol{m}_N$, the controller computes the control input $u_0 \in \text{argmax}_u J(\boldsymbol{p}_A \cup \boldsymbol{m}_N, u)$, concisely denoted by $U(\boldsymbol{p}_A \cup \boldsymbol{m}_N)$. This is expected to obtain the utility $J(\boldsymbol{p}_A \cup \boldsymbol{m}_N, u_0)$. However, the expected utility is obtained only if there is no detection error. Unfortunately, if there is a detection error, a different and potentially much lower utility is obtained.

4.3.1. Obtained Utility vs. Optimal Utility

We now quantify the actual obtained utility in presence of detection errors. Let $fp \subseteq A$ be the set of false positives, that is, sensors in fp are normal but they are marked anomalous. Since these sensors are normal, the measurements $\boldsymbol{m}_{fp}$ should have been transmitted to the controller, but due to false positives, the predictions were mistakenly transmitted. Similarly, let $fn \subseteq N$ be the set of false negatives, that is, sensors in fn are anomalous but they are marked normal. Since these sensors are anomalous, the predictions $\boldsymbol{p}_{fn}$ should have been transmitted to the controller but the measurements were mistakenly transmitted. Hence, for the control input $u_0 = U(\boldsymbol{p}_A \cup \boldsymbol{m}_N)$ computed above, the

obtained utility will actually be $J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, u_0)$. On the other hand, if there were not detection errors, the optimal control input would have been $u^* \in \text{argmax}_u J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, u)$, concisely denoted by $U(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn})$.

4.3.2. Utility Loss

To put this all together, given decisions A and N (computed given $\boldsymbol{r}$ and $\boldsymbol{\tau}$ as (3)), and the detection performance sets tp, fp, tn, and fn, the probability of occurrence of such detection error scenario is

$$\Pr(\boldsymbol{\tau}, tp, fp, tn, fn) = \prod_{s \in tp} TP_s(\tau_s) \cdot \prod_{s \in fp} FP_s(\tau_s) \cdot \prod_{s \in tn} TN_s(\tau_s) \cdot \prod_{s \in fn} FN_s(\tau_s) \,. \tag{6}$$

As discussed above, in this case, we could have obtained the optimal utility $J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}))$, but we obtained the smaller utility $J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_A \cup \boldsymbol{m}_N))$. Thus, we incurred a utility loss of

$$\Delta J = \underbrace{J^*(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn})}_{\text{Optimal Utility}} - \underbrace{J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_A \cup \boldsymbol{m}_N))}_{\text{Obtained Utility}} \,. \tag{7}$$

Hence, the expected utility loss of detection error scenario $tp \subseteq A$, $fp = A - tp$, $tn \subseteq N$, and $fn = N - tn$ is

$$C(\boldsymbol{\tau}, tp, fp, tn, fn) = \Pr(\boldsymbol{\tau}, tp, fp, tn, fn) \cdot \Delta J \,. \tag{8}$$

where $\Pr(\boldsymbol{\tau}, tp, fp, tn, fn)$ is obtained using (6) and ΔJ is obtained using (7).

4.3.3. Worst-Case Analysis

Since the sets of false positives and false negatives are not know a priori, we need to consider any possible scenario. We define the worst-case loss due to detection errors as follows.

Definition 1 (Worst-Case Detection Error Loss)**.** *Given the thresholds $\boldsymbol{\tau}$ and the residuals $\boldsymbol{r}$, the worst-case loss due to detection errors is defined as*

$$L(\boldsymbol{\tau}) = \max_{\substack{tp \subseteq A, tn \subseteq N \\ fp = A - tp \\ fn = N - tn}} C(\boldsymbol{\tau}, tp, fp, tn, fn) \,, \tag{9}$$

where $C(\boldsymbol{\tau}, tp, fp, tn, fn)$ is defined as (8), and A and N are found using (3).

4.4. Optimal Application-Aware Anomaly Detection Problem

To protect against the utility loss due to detection errors, the designer must choose the thresholds that result in the best performance with respect to the worst-case loss (9). An application-aware anomaly detector achieves this by finding the optimal thresholds $\boldsymbol{\tau}^*$ in each time step. We call this problem the Application-Aware Anomaly Detection Problem:

Definition 2 (Application-Aware Anomaly Detection Problem)**.** *Given a system model, an anomaly detector, and measured and predicted sensor values, the Application-Aware Anomaly Detection Problem is finding the optimal thresholds $\boldsymbol{\tau}^*$ that minimizes the loss (9); in other words,*

$$\boldsymbol{\tau}^* \in \operatorname*{argmin}_{\boldsymbol{\tau}} L(\boldsymbol{\tau}) \,. \tag{10}$$

If we are not able to change the thresholds at each timestep, and instead can change thresholds every T timesteps, we define

$$\bar{L}(\boldsymbol{\tau}) = \frac{1}{T}\sum_{k=1}^{T} L^k(\boldsymbol{\tau}) \,, \tag{11}$$

and then we find thresholds $\boldsymbol{\tau}^*$ that minimize the above equation. We call this problem the Static Application-Aware Anomaly Detection Problem:

Definition 3 (Static Application-Aware Anomaly Detection Problem)**.** *Given a system model, an anomaly detector, and measured and predicted sensor values, the Static Application-Aware Anomaly Detection Problem in a time period T is finding the optimal thresholds $\boldsymbol{\tau}^*$ that minimizes the loss (11):*

$$\boldsymbol{\tau}^* \in \operatorname*{argmin}_{\boldsymbol{\tau}} \bar{L}(\boldsymbol{\tau}) \,. \tag{12}$$

Clearly, (10) can be solved as a special case of (12) for $T = 1$.

5. Analysis

In this section, we solve the problems (10) and (12). First, we analyze the problem of worst-case detection error loss (9), and we prove that solving this problem is computationally challenging. We then present Algorithm 2 which is an efficient algorithm to obtain approximately optimal solutions. Second, we present Algorithm 3 to solve the application-aware detection problem (10) and obtain near-optimal thresholds. Finally, we propose Algorithm 4 to solve the problem of application-aware detection in a time period. The algorithm implements a variation of simulated annealing algorithm and finds near-optimal detection thresholds.

5.1. Algorithm for Worst-Case Detection Error Loss Problem

We begin our analysis by studying the computational complexity of finding worst-case loss due to detection errors (9). To this end, we formulate the problem of finding a worst-case loss as a decision problem.

Definition 4 (Worst-Case Detection Error Problem (Decision Version))**.** *Given a set of sensors S, detection thresholds $\boldsymbol{\tau}$, residuals $\mathbf{r}$, and desired loss L^*, determine whether there exists a detection error scenario that incurs the detection error loss of at least L^*.*

The following theorem establishes the computational complexity of finding a worst-case detection error.

Theorem 1. *Worst-Case Detection Error Problem (WCDE) is NP-Hard.*

Proof. We prove the above theorem using a reduction from a well-known NP-hard problem, the Maximum Independent Set Problem.

Definition 5 (Maximum Independent Set Problem (Decision Version))**.** *Given an undirected graph $G = (V, E)$ and a threshold cardinality k, determine whether there exists an independent set of nodes (i.e., a set of nodes such that there is no edge between any two nodes in the set) of cardinality k.*

Given an instance of the Maximum Independent Set Problem (MIS), that is, a graph $G = (V, E)$ and a threshold cardinality k, we construct an instance of the WCDE as follows:

- Let the set of sensors be $S := V$.
- Let $p_s = 0$ and $m_s = 1$ for every sensor $s \in S$.
- For every sensor $s \in S$, let $\tau_s = \epsilon$ where $\epsilon < 1$, so that $A = S$ and $N = \emptyset$.

- Let $TP_s(\tau_s) = FP_s(\tau_s) = TN_s(\tau_s) = FN_s(\tau_s) = 0.5$ for every sensor $s \in S$.
- Let the dimension of the control signal be $|S|$. For each element i of u, let $u_i \in \{0, 1\}$.
- Let the utility function be $J(\boldsymbol{w}, u) = \|\boldsymbol{w} \circ u\|_1$ if the non-zero elements in $\boldsymbol{w}$ form a non-empty independent set, and $-\|u\|_1$ otherwise.
- Finally, let the threshold loss be $L^* := \left(\frac{1}{2}\right)^{|S|} k$.

Clearly, the above reduction can be performed in polynomial time. Hence, it remains to show that the constructed instance of WCDE has a solution *if and only if* the given instance of MIS does.

MIS then WCDE. First, suppose that MIS has a solution, that is, there exists an independent set I of k nodes. We claim that the set $fp = I$ and $tp = S - I$ is a solution to WCDE. We have

$$\Delta J = J^*(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp}) - J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp}, U(\boldsymbol{p}_A))) = J^*(\boldsymbol{m}_{fp}) - J(\boldsymbol{m}_{fp}, <0>_{i\in 1}^{|S|}) = \left\|\boldsymbol{m}_{fp}\right\|_1 - 0 = k$$

Since $\Pr(\boldsymbol{\tau}, tp, fp, tn, fn) = \left(\frac{1}{2}\right)^{|S|}$ for any given sets of detection error, we obtain $L(\boldsymbol{\tau}) = \left(\frac{1}{2}\right)^{|S|} \cdot k$.

Not MIS then Not WCDE. Second, suppose that MIS has no solution, that is, every set of at least k nodes is non-independent. Then, we have that $J(\boldsymbol{w}, u) < k$ for every $\boldsymbol{w}$; otherwise, there would exist a set of at least k nodes in I that are independent of each other, which would contradict our supposition. Then, since $\Pr(\boldsymbol{\tau}, tp, fp, tn, fn) = \left(\frac{1}{2}\right)^{|S|}$, we conclude $L(\boldsymbol{\tau}) < \left(\frac{1}{2}\right)^{|S|} \cdot k$. □

Algorithm 2 Algorithm for Computing Worst-Case Loss.

```
1:  function WORST_LOSS(τ, m, p)
2:      for all s ∈ S do
3:          (|m_s − p_s| ≤ τ_s) ? N ← N ∪ {s} : A ← A ∪ {s}
4:      end for
5:      tp ← A, fp ← ∅
6:      tn ← N, fn ← ∅
7:      L* ← 0
8:      while tp ≠ ∅ or tn ≠ ∅ do
9:          (C_i, i) ← max_{i⊆tp} C(τ, tp \ i, fp ∪ {i}, tn, fn)
10:         (C_j, j) ← max_{j⊆tn} C(τ, tp, fp, tn \ j, fn ∪ {j})
11:         if C_i < L_j then
12:             C ← C_j
13:             tn ← tn \ j
14:             fn ← fn ∪ {j}
15:         else
16:             C ← C_i
17:             tp ← tp \ i
18:             fp ← fp ∪ {i}
19:         end if
20:         if C* < C then
21:             C* ← C
22:         else
23:             return C*
24:         end if
25:     end while
26:     return C*
27: end function
```

We present Algorithm 2 which uses a greedy approach to obtain the worst-case loss due to detection errors. The algorithm starts considering a scenario of perfect detection, that is, $tp = A$,

$fp = \emptyset$, $tn = N$ and $fn = \emptyset$. In each iteration, the algorithm moves an element from either tp or tn to respectively fp or fn that maximally increases the utility loss. If no such element exists, the algorithm terminates with the best solution found so far.

The runtime of Algorithm 2 depends on the function J, which depends on the considered application. If there is an oracle that computes $U(\boldsymbol{w})$ and $J(\boldsymbol{w}, U(\boldsymbol{w}))$ in constant time, the runtime of Algorithm 2 is linear with respect to $|S|$. That is, the runtime of Algorithm 2 is $\mathcal{O}(|S|)$.

5.2. Algorithm for Application Aware Anomaly Detection Problem

To solve the AAAD problem, we first prove the following lemma. The lemma shows that the problem is equal to the problem of selecting a set of normal sensors N and a set of anomalous sensors A, which has a much smaller search space than the original problem.

Lemma 1. *For sensor s with residual r_s, the optimal threshold with respect to (10) satisfies $\tau_s \in \{0, r_s, r_s^+, M\}$.*

Proof. We need to prove that for sensor s with residual r_s, the optimal threshold with respect to (10) is in the set $\{0, r_s, r_s^+, M\}$. First, let us recall that the optimal threshold is

$$\boldsymbol{\tau}^* \in \operatorname*{argmin}_{\boldsymbol{\tau}} \max_{\substack{tp \subseteq A, tn \subseteq N \\ fp = A - tp \\ fn = N - tn}} \Pr(\boldsymbol{\tau}, tp, fp, tn, fn) \cdot \Delta J \ ,$$

where $\Delta J = J^*(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}) - J(\boldsymbol{p}_{tp} \cup \boldsymbol{m}_{fp} \cup \boldsymbol{m}_{tn} \cup \boldsymbol{p}_{fn}, U(\boldsymbol{p}_A \cup \boldsymbol{m}_N))$.

Suppose there exists a set of optimal thresholds $\boldsymbol{\tau}^*$ such that some of its elements are not in the set mentioned above. Let s be one such sensor, that is, $\tau_s^* \notin \{0, r_s, r_s^+, M\}$. First, let $0 < \tau_s^* < r_s$. Clearly, $\Delta J(\boldsymbol{\tau}^*) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{0\}) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{r_s\})$. Then, we write $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{TP_s(\tau_s')}{TP_s(\tau_s^*)} > 1$ if $\tau_s' = 0$, and so τ_s^* cannot be the optimal threshold if s is in the set of true positives. Also, $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{FP_s(\tau_s')}{FP_s(\tau_s^*)} > 1$ if $\tau_s' = r_s$, and so τ_s^* cannot be the optimal threshold if s is in the set of false positives either. Second, let $r_s^+ < \tau_s^* < M$. Again, we have $\Delta J(\boldsymbol{\tau}^*) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{r_s^+\}) = \Delta J(\boldsymbol{\tau}_{-s} \cup \{M\})$. Then, we write $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{TN_s(\tau_s')}{TN_s(\tau_s^*)} > 1$ if $\tau_s' = M$. Also, $\frac{\Pr(\boldsymbol{\tau}', tp, fp, tn, fn)}{\Pr(\boldsymbol{\tau}^*, tp, fp, tn, fn)} = \frac{FN_s(\tau_s')}{FN_s(\tau_s^*)} > 1$ if $\tau_s' = r_s^+$. This means that τ_s^* cannot be the optimal threshold if s is in the set of true negatives or false negatives either. This contradicts our supposition, and thus, $\tau_s \notin \{0, r_s, r_s^+, M\}$ can never be correct. This concludes our proof. □

Algorithm 3 Algorithm for Design of Application-Aware Detector.

```
function APPLICATION_AWARE(m, p)
    N ← S, A ← ∅
    L* ← ∞
    while A ≠ S do
        (L, s) ← argmin_{s∈N} WORST_LOSS(A ∪ {s}, N \ {s}, m, p)
        if L* < L then
            L* ← L
            A ← A ∪ {s}
            N ← N \ {s}
        else
            return L*
        end if
    end while
    return L*
end function
```

Following the above lemma, we present Algorithm 3 to obtain application-aware detection thresholds. The algorithm begins by initializing all sensors as normal, that is, $N = S$ and $A = \emptyset$. In each iteration, the algorithm moves a sensor from N to A, which maximally decreases the worst-case loss. To compute the worst-case loss, Algorithm 2 is used.

Similar to the previous algorithm, the running time depends on the function J and the considered application. If there is an oracle that returns $U(\boldsymbol{w})$ and $J(\boldsymbol{w}, U(\boldsymbol{w}))$ in constant time, the runtime of Algorithm 3 is $\mathcal{O}(|S|^2)$.

5.3. Algorithm for Application-Aware Anomaly Detection in a Time Period

We present Algorithm 4 which solves the problem of application-aware detection in a time period T (2). The algorithm is based on a variation of simulated annealing algorithm, and finds near-optimal thresholds $\boldsymbol{\tau}$. The idea is to start with an arbitrary solution $\boldsymbol{\tau}$ and improving it iteratively. In each iteration, we generate a new candidate solution $\boldsymbol{\tau}'$ in the neighborhood of $\boldsymbol{\tau}$. If the candidate solution $\boldsymbol{\tau}'$ is better in minimizing the loss, then the current solution is replaced with the new one. However, if $\boldsymbol{\tau}'$ increases the loss, the new solution replaces the current solution with only a small probability. This probability depends on the difference between the two solutions in terms of loss as well as a temperature parameter which is a decreasing function of the number of iterations. These random replacements decreases the likelihood of getting stuck in a local minimum.

Algorithm 4 Algorithm for Design of the Application-Aware in a Time Period.

Input: $\boldsymbol{m}, \boldsymbol{p}$
Initialize: $\boldsymbol{\tau}$, $n \leftarrow 1$, T_0
$L(\boldsymbol{\tau}) \leftarrow$ WORST_LOSS$(\boldsymbol{\tau}, \boldsymbol{m}, \boldsymbol{p})$
while $n \leq n_{\max}$ **do**
  $\boldsymbol{\tau}' \leftarrow$ PERTURB$(\boldsymbol{\tau}, n)$
  $L(\boldsymbol{\tau}') \leftarrow$ WORST_LOSS$(\boldsymbol{\tau}', \boldsymbol{m}, \boldsymbol{p})$
  $c \leftarrow e^{(L(\boldsymbol{\tau}') - L(\boldsymbol{\tau}))/T}$
  if $(L(\boldsymbol{\tau}') < L(\boldsymbol{\tau})) \vee (\texttt{rand}(0,1) \leq c)$ **then**
    $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau}'$, $L(\boldsymbol{\tau}) \leftarrow L(\boldsymbol{\tau}')$
  end if
  $T \leftarrow T_0 \cdot e^{-\beta n}$
  $n \leftarrow n + 1$
end while
return $\boldsymbol{\tau}$

In Algorithm 4, PERTURB$(\boldsymbol{\tau}, n)$ defines the neighborhood of $\boldsymbol{\tau}$ in the nth iteration, from which $\boldsymbol{\tau}'$ is randomly sampled. More specifically, PERTURB$(\boldsymbol{\tau}, n)$ means that each τ_s in $\boldsymbol{\tau}$ is replaced by $\tau_s' = \tau_s + \Delta\tau_s$. Here, for each $s \in S$, $\Delta\tau_s$ is randomly picked from the uniform distribution over $\left[-\alpha\left(\frac{n_{\max}-n}{n_{\max}}\right), \alpha\left(\frac{n_{\max}-n}{n_{\max}}\right)\right]$ for some $\alpha \in \mathcal{R}_+$. Moreover, since τ_s' is nonnegative, we replace it with 0 if $\tau_s' < 0$.

6. Special Cases

In this section, we consider two special cases for the AAAD problem (10). The first special case is single detector, which means that either there is a single detector in the system or each detector is optimized independently of other detectors. The second special case is detectors with equal thresholds, where there are multiple detectors that have the same thresholds.

6.1. Single Detector

Consider a scenario where $|S| = 1$. This means that either there is a single detector in the system, or each detector is optimized independently and irrespective of other detectors. Let $S = \{a\}$ be the

considered sensor, and let $r_a = |p_a - m_a|$ be the residual of the sensor at a timestep. First, we consider a threshold τ'_a where $r_a > \tau'_a$ for this sensor. This threshold results in a detection alert, and so the set of marked-anomalous sensors becomes $A = \{a\}$ and the set of marked-normal sensors becomes $N = \emptyset$. Next, to find the worst-case detection error loss (9) for this threshold, there are two possibilities for tp: (1) $tp = \{a\}$ and $fp = \emptyset$, and (2) $tp = \emptyset$ and $fp = \{a\}$. For $tp = \{a\}$, i.e., no detection error, we can write

$$C(\tau'_a, \{a\}, \emptyset, \emptyset, \emptyset) = TP(\tau'_a) \cdot \Big(J(p_a, U(p_a)) - J(p_a, U(p_a)) \Big) = 0 \,.$$

For the second scenario, i.e., $tp = \emptyset$ and $fp = \{a\}$, we should have used the measurement m_a but we used the prediction p_a, and so the expected utility loss is

$$C(\tau'_a, \emptyset, \{a\}, \emptyset, \emptyset) = FP(\tau'_a) \cdot \Big(J(m_a, U(m_a)) - J(m_a, U(p_a)) \Big) \,.$$

Therefore, the worst-case utility loss for the threshold τ'_a, where $r_a > \tau'_a$, is obtained using

$$L(\tau'_a) = \max C(\tau'_a, \emptyset, \{a\}, \emptyset, \emptyset) = C(r_a^-, \emptyset, \{a\}, \emptyset, \emptyset)$$

Next, we consider a threshold τ''_a such that $r_a \leq \tau''_a$. This threshold results in no detection alert, and so $A = \emptyset$ and $N = \{a\}$. To compute the worst-case detection error loss, there are two possibilities for detection error: (1) $tn = \{a\}$ and $fn = \emptyset$, and (2) $tn = \emptyset$ and $fn = \{a\}$. Similar to the above scenario, for the first case which corresponds to no detection error, we obtain $\Delta J = 0$ and so $C(\tau''_a, \emptyset, \emptyset, \{a\}, \emptyset) = 0$. For the second case, we obtain

$$C(\tau''_a, \emptyset, \emptyset, \emptyset, \{a\}) = FN(\tau''_a) \cdot \Big(J(p_a, U(p_a)) - J(p_a, U(m_a)) \Big) \,.$$

Therefore, the worst-case detection error loss for the threshold τ''_a, where $r_a \leq \tau''_a$, is

$$L(\tau''_a) = \max C(\tau''_a, \emptyset, \emptyset, \emptyset, \{a\}) = C(r_a, \emptyset, \emptyset, \emptyset, \{a\}) \,.$$

The application-aware detector selects the threshold $\tau^*_a \in \{r_a^-, r_a\}$ that solves

$$\min\big(C(r_a^-, \emptyset, \{a\}, \emptyset, \emptyset), C(r_a, \emptyset, \emptyset, \emptyset, \{a\}) \big) \,.$$

In other words, the optimal threshold is

$$\tau_a = \begin{cases} r_a^- & \text{if } C(r_a^-, \emptyset, \{a\}, \emptyset, \emptyset) \leq C(r_a, \emptyset, \emptyset, \emptyset, \{a\}) \\ r_a & \text{otherwise} \end{cases} \,. \tag{13}$$

6.2. Detectors with Equal Thresholds

We consider a case where all detectors have equal thresholds. Let $\bar{\tau}$ represent this threshold value, that is, $\bar{\tau} = \tau_1 = \ldots = \tau_d$. Next, let $r_1, r_2, \ldots, r_d$ be the residual values. The result below is a direct consequence of Lemma 1.

Corollary 1. *The optimal threshold $\bar{\tau}$ for detectors with equal thresholds belongs to the following set*

$$\texttt{SolutionSpace} = \{0, r_1^-, r_1^+, r_2^-, r_2^+ \ldots, r_d^-, r_d^+, M\} \,.$$

Based on the above corollary, since the solution space is finite, we can find the optimal thresholds by a linear-time search, as presented by Algorithm 5.

Algorithm 5 Application-Aware Detector for Equal Thresholds.

Input: $\boldsymbol{m}$, $\boldsymbol{p}$
$\boldsymbol{r} \leftarrow |\boldsymbol{m} - \boldsymbol{p}|$
for all $\bar{\tau} \in$ `SolutionSpace` **do**
    $(L(\bar{\tau}), \bar{\tau}) \leftarrow$ WORST_LOSS$(\bar{\tau}, \boldsymbol{m}, \boldsymbol{p})$
end for
$\bar{\tau}^* \leftarrow \text{argmin}(L(\bar{\tau}), \bar{\tau})$
return $\bar{\tau}^*$

7. Evaluation

In the preceding sections, we presented general computational results and general-purpose algorithms for the AAAD problem. In this section, to demonstrate the practical application of our AAAD framework, we provide numerical results on our running example. In particular, we apply our approach to a case study of max-pressure control of traffic signals in a traffic network. First, we construct regression-based anomaly detectors for traffic sensors, and we generate the trade-off curves for their performance. Then, we implement the AAAD approach, and evaluate its performance compared to a baseline "application-unaware" detector configuration. Throughout this section, we use SUMO (Simulation of Urban MObility), which is a micro simulator for traffic applications [27].

7.1. Traffic Network

Consider a traffic network in a 3-by-3 grid with a total of 9 intersections, as show in Figure 4. We perform our numerical evaluation in a simulated environment, where we have accurate ground truth regarding sensor faults. Each intersection connects 4 standard two-way lanes with four possible movements {EW, WE, NS, SN}, as shown in Figure 5. Traffic volume of each movement is monitored by the set of sensors $S = \{s_{EW}, s_{WE}, s_{NS}, s_{SN}\}$. The sensors send traffic measurements $\boldsymbol{m} = \{m_{EW}, m_{WE}, m_{NS}, m_{SN}\}$ at each timestep. Each traffic signal has two phases $\Phi = \{\phi_{\{EW,WE\}}, \phi_{\{NS,SN\}}\}$. The max-pressure controller computes the optimal stage $u^* \in \Phi$ using (2).

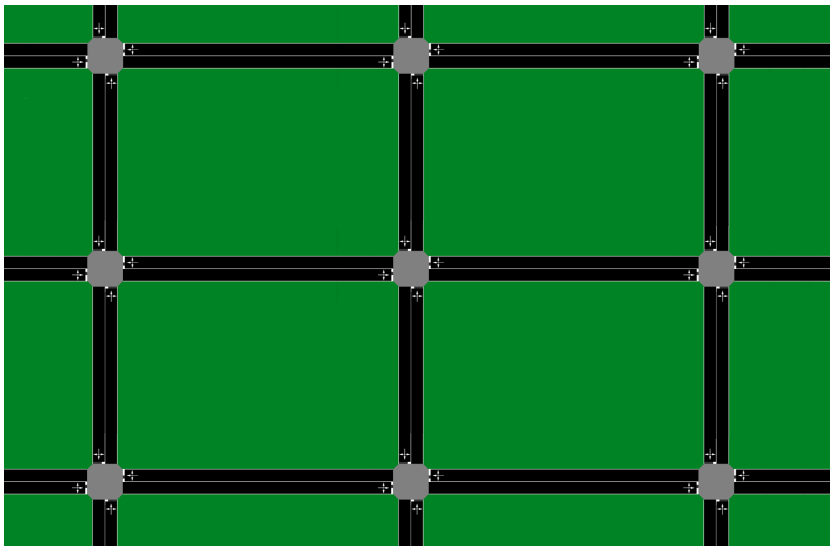

Figure 4. The 3-by-3 grid.

The utility (i.e., pressure-release) function of the traffic network can be written as sum of the pressure-release of each individual intersection, and for each intersection, the utility depends only on its corresponding lanes. This means that maximizing the pressure-release of the traffic network is equal to maximizing the pressure-release of individual intersections, and so the application-aware

threshold of each intersection can be designed independently of other intersections. Based on this observation, in what follows, we discuss how the detector is designed for an intersection and then extend it to all intersections.

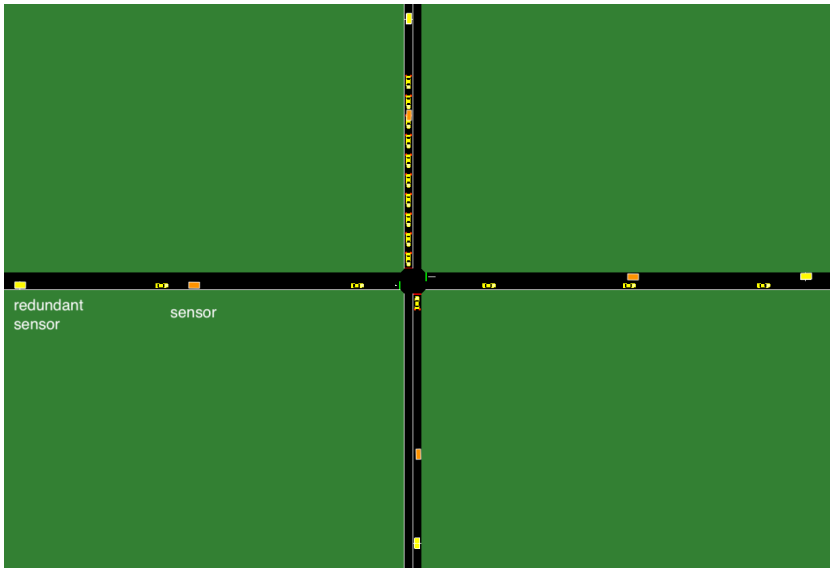

Figure 5. Each intersection in the 3-by-3 grid. For illustration, a critical and a redundant sensor are shown in the figure. For each lane, a second redundant sensor is placed with twice as much distance as the distance between the first redundant sensor and the critical sensor. All 8 total redundant sensors are used to predict the value of a sensor.

7.1.1. Anomaly Model

Traffic measurements may be anomalous due to failures or other undesired events. To simulate the negative effects of anomalies on the system, we consider several realistic anomaly models [28],

1. *Overcount*: Additive error equal to 3% to 7% of the actual values, i.e., $e_s(k) = u_s a_s(k)$ and $0.03 \leq u_s \leq 0.07$.
2. *Undercount*: Subtractive error equal to 7% to 13% of the actual values, i.e., $e_s(k) = u_s a_s(k)$ and $-0.13 \leq u_s \leq -0.07$.
3. *Gaussian Noise*: Error with zero mean and standard deviation $\sigma = 15$ to $\sigma = 35$, i.e., $e_s(k) \sim \mathcal{N}(0, \sigma^2)$ and $15 \leq \sigma \leq 35$.

7.2. Regression-Based Detector and Trade-Off Curves

To protect the system against anomalies, we construct anomaly detectors. We suppose there are 8 (2 on each side) redundant sensors that are adjacent to the four critical sensors mentioned above. We use the values of these sensors to design regression-based anomaly detectors for the critical sensors. As discussed in the preceding sections, such detector consists of two main components: (1) Predictor and (2) Statistical Test.

7.2.1. Predictor

We collect simulation data that represents the traffic behavior under normal operation. We simulate the network for 4 h considering a Poisson distribution as the demand for each movement. We collect sensor measurements in 10-s aggregates. The data from the first 2 h is used to train the predictors, and the data from the remaining 2 h is used to obtain the trade-off curves. Following our previous discussion, for each predictor, we use the current value of the 8 redundant sensors as the features. Then, we train the predictor using linear regression algorithm. We obtain the performance metrics $MSE_{train} = 2.14$ and $MSE_{test} = 2.87$. Please note that more complex regression algorithms

(e.g., Gaussian Processes [24]) can be used as well; however, we obtained satisfactory result with a simple linear regression model.

7.2.2. Trade-off Curve

To generate the trade-off curve, first, we simulate the anomalies on the test data, and evaluate the performance of the detector by counting the number of true positives and false negatives. Similarly, we simulate the system under normal operation and evaluate the performance of the detector to obtain the number of true negatives and false positives. We repeat the steps while varying the detection threshold in order to obtain the trade-off curve (i.e., true positive probability as a function of false positive probability). Figure 6 shows the resulting trade-off curve.

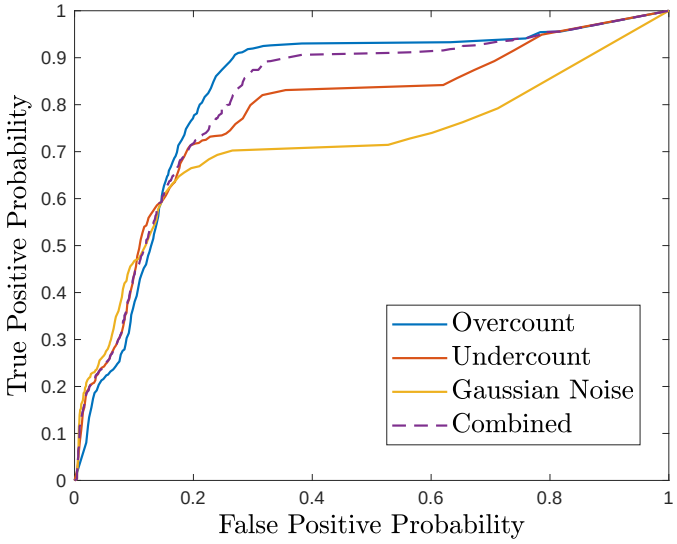

Figure 6. Trade-off between true positive and false positive errors for s_{EW}.

7.3. Application-Aware Anomaly Detector

Given the trade-off curve, we implement the application-aware anomaly detector by finding the detection thresholds that minimize the worst-case expected utility loss. First, we show how the optimal threshold can be computed at each single timestep. Then, we present the results.

7.3.1. Computing the Optimal Threshold

To show how the optimal threshold is computed at each timestep, suppose $m_{NS} = p_{NS} = 15$ at a given timestep. Further, suppose there is no traffic on SN and WE. If $w_{EW} \geq 15$, the max-pressure controller selects the stage $u = \phi_{\{EW,WE\}}$, and otherwise, it selects $u = \phi_{\{NS,SN\}}$. Next, consider the following scenarios for m_{EW} and p_{EW}:

1. $m_{EW} = 30$ and $p_{EW} = 10$. In this case, the threshold that solves the following equation is the optimal threshold: $\min\big(C(20^-,\emptyset,\{s_{EW}\},\emptyset,\emptyset), C(20,\emptyset,\emptyset,\emptyset,\{s_{EW}\})\big)$, where $C(20^-,\emptyset,\{s_{EW}\},\emptyset,\emptyset) = FP(20^-)\cdot\big(J^*(30) - J(30,U(10))\big) = 30-15 = 15$, and $C(20,\emptyset,\emptyset,\emptyset,\{s_{EW}\}) = FN(20)\cdot\big(J^*(10)) - J(10,U(30))\big) = 15-10 = 5$. Thus, the optimal threshold is $\tau^* = 20$. To see why this is the optimal threshold, note that if $\tau = 20$, then there will be no detection alert and the measurement ($m_{EW} = 30$) will be used in computing the optimal control input (which is EW). If this is incorrect (due to a false negative), then $p_{SE} = 10$ is the actual value. In this case, the pressure-release of $\phi_{\{NS,SN\}}$ will be 10. In the perfect detection case (no detection error), we could have obtained 15 by selecting NS. Thus, the detection error loss is $15-10 = 5$. On the other hand, if $\tau = 20^-$, the prediction will be used by the controller.

This results in the movement NS being selected. However, if there is a false positive, which means $m_{EW} = 30$ is the actual value, then EW should have been selected as the optimal stage, which could have obtained a utility of 30. Thus, the utility loss in this case is $30 - 15 = 15$, which is larger than the utility loss for the other threshold.

2. $m_{EW} = 30$, $p_{EW} = 10$, and $a_{NS} = 25$. The application-aware detector computes the minimum of $FP^{-}(20) \cdot \left(J^{*}(30) - J(30, U(10))\right) = 30 - 25 = 5$, and $FN(20) \cdot \left(J^{*}(10) - J(10, U(30))\right) = 25 - 10 = 15$. In this case, the threshold $\tau = 20^{-}$ is the optimal threshold.
3. $m_{EW} = 30$, $p_{EW} = 25$, and $a_{NS} = 15$. The application-aware detector computes $FP(20^{-}) \cdot \left(J^{*}(30) - J(30, U(25))\right) = 30 - 30 = 0$ and $FN(20) \cdot \left(J^{*}(25) - J(25, U(30))\right) = 25 - 25 = 0$, which means that both thresholds are optimal. The same results can be obtained if $a_{NS} \geq m_{EW}$ and $a_{NS} \geq p_{EW}$.

7.3.2. Comparison

We now compare our application-aware detector configuration to a baseline detector configuration, which does not take into account the underlying CPS application. For a fair comparison, the threshold of the baseline detector is selected such that it attains the same false-alarm probability as the application-aware anomaly detector. That is, we first calculate the total number of false alarms for the application-aware detector configuration, and then select the threshold that obtains the same false-alarm probability for the baseline configuration. In our numerical example, in 2 h of evaluation time during which each of the anomalies may occur with probability of 0.05, each application-aware detector had on average a false-alarm probability of 0.047. The threshold values for the application-aware detector varied from 1.3 to 26.5 with a mean of 5.6. The threshold for the baseline detector was selected to be 3.9.

Figure 7 shows the pressure-release comparison between the two cases during the 2-h interval. Each tick in the figure aggregates the results from 12 min (i.e., 72 timesteps). Based on the results, the application-aware detector performs better than the baseline detector in most cases. The baseline detector performs only slight better in the aggregated timestep at 0.8, which could be due to detection errors by the application-aware detector. However, in the rest of the 2-h period, the proposed detector perform significantly better.

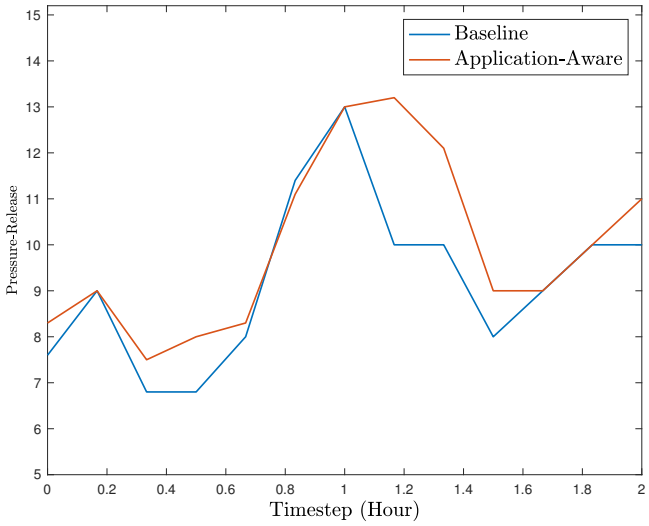

Figure 7. Utility (i.e., pressure-release) during a 2-h interval.

8. Conclusions

We presented the application-aware anomaly detection framework for detecting anomalies in sensor measurements in cyber-physical systems. An application-aware anomaly detector configures

itself such that the application performance in the presence of detection errors is as close as possible to the performance that could have been obtained if there were no detection errors. We formulated and studied the problem of optimal, application-aware configuration of an existing detector. We evaluated our result using a case study of real-time control of traffic signals, and showed that our application-aware detector configuration significantly outperforms the baseline.

Author Contributions: The individual contributions of the authors are as follows: "Conceptualization, A.G., A.L. and X.K; Methodology, A.G., A.L. and X.K; Software, A.G.; Formal Analysis, A.G. and A.L.; Writing—Original Draft Preparation, A.G.; Writing—Review & Editing, X.K. and A.L.; Visualization, A.G.; Project Administration, X.K.; Funding Acquisition, X.K."

Funding: This research is supported in part by the National Science Foundation under award CNS-1238959 and the National Institute of Standards and Technology under award 70NANB17H266. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Urbina, D.I.; Giraldo, J.A.; Cardenas, A.A.; Tippenhauer, N.O.; Valente, J.; Faisal, M.; Ruths, J.; Candell, R.; Sandberg, H. Limiting the impact of stealthy attacks on industrial control systems. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), Vienna, Austria, 24–28 October 2016; ACM: New York, NY, USA, 2016; pp. 1092–1105.
2. Laszka, A.; Potteiger, B.; Vorobeychik, Y.; Amin, S.; Koutsoukos, X. Vulnerability of Transportation Networks to Traffic-Signal Tampering. In Proceedings of the 2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS), Vienna, Austria, 11–14 April 2016; pp. 1–10.
3. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv. (CSUR)* **2009**, *41*, 15, doi:10.1145/1541880.1541882.
4. Laszka, A.; Abbas, W.; Sastry, S.S.; Vorobeychik, Y.; Koutsoukos, X. Optimal Thresholds for Intrusion Detection Systems. In Proceedings of the 3rd Annual Symposium and Bootcamp on the Science of Security (HotSoS), Pittsburgh, PA, USA, 19–21 April 2016; pp. 72–81.
5. Ghafouri, A.; Abbas, W.; Laszka, A.; Vorobeychik, Y.; Koutsoukos, X. Optimal Thresholds for Anomaly-Based Intrusion Detection in Dynamical Environments. In Proceedings of the 7th International Conference on Decision and Game Theory for Security (GameSec), New York, NY, USA, 2–4 November 2016; Volume 9996, pp. 415–434.
6. Ostfeld, A.; Uber, J.G.; Salomons, E.; Berry, J.W.; Hart, W.E.; Phillips, C.A.; Watson, J.P.; Dorini, G.; Jonkergouw, P.; Kapelan, Z.; et al. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *J. Water Resour. Plan. Manag.* **2008**, *134*, 556–568.
7. Hu, F.; Lu, Y.; Vasilakos, A.V.; Hao, Q.; Ma, R.; Patil, Y.; Zhang, T.; Lu, J.; Li, X.; Xiong, N.N. Robust cyber–physical systems: Concept, models, and implementation. *Future Gener. Comput. Syst.* **2016**, *56*, 449–475.
8. Mo, Y.; Sinopoli, B. On the performance degradation of cyber-physical systems under stealthy integrity attacks. *IEEE Trans. Autom. Control* **2016**, *61*, 2618–2624.
9. Goh, J.; Adepu, S.; Tan, M.; Lee, Z.S. Anomaly detection in cyber physical systems using recurrent neural networks. In Proceedings of the 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), Singapore, 12–14 January 2017; pp. 140–145.
10. Kosek, A.M. Contextual anomaly detection for cyber-physical security in Smart Grids based on an artificial neural network model. In Proceedings of the 2016 Joint Workshop on Cyber-Physical Security and Resilience in Smart Grids (CPSR-SG), Vienna, Austria, 12 April 2016; pp. 1–6.
11. Krishnamurthy, S.; Sarkar, S.; Tewari, A. Scalable anomaly detection and isolation in cyber-physical systems using bayesian networks. In Proceedings of the ASME 2014 Dynamic Systems and Control Conference, American Society of Mechanical Engineers, New York, NY, USA, 30 October 2014; pp. V002T26A006–V002T26A006.

12. Jones, A.; Kong, Z.; Belta, C. Anomaly detection in cyber-physical systems: A formal methods approach. In Proceedings of the 2014 IEEE 53rd Annual Conference on Decision and Control (CDC), Los Angeles, CA, USA, 15–17 December 2014; pp. 848–853.
13. Kong, Z.; Jones, A.; Belta, C. Temporal logics for learning and detection of anomalous behavior. *IEEE Trans. Autom. Control* **2017**, *62*, 1210–1222.
14. Chibani, A.; Chadli, M.; Ding, S.X.; Braiek, N.B. Design of robust fuzzy fault detection filter for polynomial fuzzy systems with new finite frequency specifications. *Automatica* **2018**, *93*, 42–54.
15. Chibani, A.; Chadli, M.; Braiek, N.B. A finite frequency approach to H∞ filtering for T–S fuzzy systems with unknown inputs. *Asian J. Control* **2016**, *18*, 1608–1618.
16. Li, L.; Chadli, M.; Ding, S.; Qiu, J.; Yang, Y. Diagnostic Observer Design for T–S Fuzzy Systems: Application to Real-Time-Weighted Fault-Detection Approach. *IEEE Trans. Fuzzy Syst.* **2018**, *26*, 805–816.
17. Mitchell, R.; Chen, I.R. A survey of intrusion detection techniques for cyber-physical systems. *ACM Comput. Surv.* **2014**, *46*, 55.
18. Kleinmann, A.; Wool, A. Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems. *ACM Trans. Intell. Syst. Technol. (TIST)* **2017**, *8*, 55.
19. Lu, X.Y.; Varaiya, P.; Horowitz, R.; Palen, J. Faulty loop data analysis/correction and loop fault detection. In Proceedings of the 15th World Congress on Intelligent Transport Systems, New York, NY, USA, 16–20 November 2008.
20. Zygouras, N.; Panagiotou, N.; Zacheilas, N.; Boutsis, I.; Kalogeraki, V.; Katakis, I.; Gunopulos, D. Towards Detection of Faulty Traffic Sensors in Real-Time. In Proceedings of the 2nd International Conference on Mining Urban Data (MUD), Lille, France, 11 July 2015; pp. 53–62.
21. Robinson, S.P. The Development and Application of an Urban Link Travel Time Model Using Data Derived from Inductive Loop Detectors. Ph.D. Thesis, University of London, London, UK, 2006.
22. Varaiya, P. Max pressure control of a network of signalized intersections. *Transp. Res. Part C Emerging Technol.* **2013**, *36*, 177–195.
23. Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; Wang, F.Y. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 865–873.
24. Ghafouri, A.; Laszka, A.; Dubey, A.; Koutsoukos, X. Optimal detection of faulty traffic sensors used in route planning. In Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE), Pittsburgh, PA, USA, 18–21 April 2017.
25. Min, W.; Wynter, L. Real-time road traffic prediction with spatio-temporal correlations. *Transp. Res. Part C Emerging Technol.* **2011**, *19*, 606–616.
26. Basseville, M.; Nikiforov, I.V. *Detection of Abrupt Changes: Theory and Application*; Prentice Hall: Englewood Cliffs, NJ, USA, 1993; Volume 104.
27. Behrisch, M.; Bieker, L.; Erdmann, J.; Krajzewicz, D. SUMO–simulation of urban mobility: an overview. In Proceedings of the 3rd International Conference on Advances in System Simulation (SIMUL), Barcelona, Spain, 23–28 October 2011.
28. Widhalm, P.; Koller, H.; Ponweiser, W. Identifying faulty traffic detectors with Floating Car Data. In Proceedings of the 2011 IEEE Forum on Integrated and Sustainable Transportation System (FISTS), Vienna, Austria, 29 June–1 July 2011.