Optimal Detection of Faulty Traffic Sensors Used in Route Planning

Amin Ghafouri
Vanderbilt University
Nashville, Tennessee 37212
amin.ghafouri@vanderbilt.edu

Aron Laszka
Vanderbilt University
Nashville, Tennessee 37212
aron.laszka@vanderbilt.edu

Abhishek Dubey
Vanderbilt University
Nashville, Tennessee 37212
abhishek.dubey@vanderbilt.edu

Xenofon Koutsoukos
Vanderbilt University
Nashville, Tennessee 37212
xenofon.koutsoukos@vanderbilt.edu

ABSTRACT

In a smart city, real-time traffic sensors may be deployed for various applications, such as route planning. Unfortunately, sensors are prone to failures, which result in erroneous traffic data. Erroneous data can adversely affect applications such as route planning, and can cause increased travel time. To minimize the impact of sensor failures, we must detect them promptly and accurately. However, typical detection algorithms may lead to a large number of false positives (i.e., false alarms) and false negatives (i.e., missed detections), which can result in suboptimal route planning. In this paper, we devise an effective detector for identifying faulty traffic sensors using a prediction model based on Gaussian Processes. Further, we present an approach for computing the optimal parameters of the detector which minimize losses due to false-positive and false-negative errors. We also characterize critical sensors, whose failure can have high impact on the route planning application. Finally, we implement our method and evaluate it numerically using a real-world dataset and the route planning platform OpenTripPlanner.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; *Dependable and fault-tolerant systems and networks*; • **Theory of computation →** *Gaussian processes*;

KEYWORDS

fault detection, cyber-physical systems, smart city, route planning

ACM Reference format:
Amin Ghafouri, Aron Laszka, Abhishek Dubey, and Xenofon Koutsoukos. 2017. Optimal Detection of Faulty Traffic Sensors Used in Route Planning. In *Proceedings of The 2nd Workshop on Science of Smart City Operations and Platforms Engineering, Pittsburgh, PA USA, 21 April 2017 (SCOPE 2017),* 6 pages.
DOI: http://dx.doi.org/10.1145/3063386.3063767

1 INTRODUCTION

In smart cities, real-time traffic sensors may be deployed for various applications. However, sensors are prone to failures, which result in erroneous traffic data. Erroneous data can adversely affect the performance of applications. To minimize the impact of sensor failures, we must detect them promptly and with high accuracy. However, typical detection algorithms may lead to a large number of false positives and false negatives, which can result in suboptimal performance.

Anomaly detection of faulty traffic sensors has been studied in the literature. Typical approaches include using data-driven methods that incorporate historical and real-time data to detect anomalies [10], [18], [14], [16]. However, existing approaches may result in high performance-losses in traffic applications, mainly due to false-positive (FP) and false-negative (FN) errors. In order to minimize the losses, it is desirable to reduce the FP and FN rates as much as possible. But, there exists a trade-off between them, which can be changed through a detection threshold. To address this, it is necessary to take into account the traffic application when designing anomaly detectors, and quantify the losses in the traffic application caused by the FP and FN errors. By selecting the right detection threshold, the performance losses caused by FPs and FNs can be minimized.

In this paper, we study the problem of finding optimal thresholds for anomaly detection of faulty traffic sensors, considering route planning as the application of interest. The objective is to select the optimal thresholds of anomaly detectors in order to optimize the performance of the route planning application in the presence of faulty sensors. We devise an effective detector for identifying faulty traffic sensors using a prediction model based on Gaussian Processes. Further, we present an approach for computing the optimal parameters of the detector which minimize losses due to false-positive and false-negative errors. We also characterize critical sensors, whose failure can have high impact on the traffic application. Finally, we implement our method and evaluate it numerically using a real-world dataset and the route planning platform OpenTripPlanner [11]. Our evaluation results show that the proposed strategy successfully minimizes the performance loss and identifies the critical sensors.

The remainder of this paper is organized as follows. In Section 2, we present the background for route planning and Gaussian Process regression. In Section 3, we introduce the system model. In Section 4, we define a notion of optimal detection, present a method to obtain near-optimal thresholds, and define critical sensors. In Section 5, we implement our method and evaluate it numerically. In Section 6, we discuss related work. Concluding remarks are presented in Section 7.

2 PRELIMINARIES

2.1 Route Planning

Let $G = (V, E)$ be a directed graph with a set V of vertices and a set E of arcs. Each arc $(u, v) \in E$ has an associated nonnegative cost $c(u, v)$. The cost (i.e., length) of a path is the sum of the costs of its arcs. In the point-to-point shortest path problem, one is given as input the graph G, a query $q = (o, d)$, where $o \in V$ is an origin and $d \in V$ is a destination, and the objective is to find a minimum-cost (i.e., shortest) path from o to d in G. In the many-to-many shortest path problem, a set of queries Q is given, and the goal is to find the minimum-cost path for each query $q = (o, d) \in Q$.

There exist many route planning algorithms that compute optimal solutions in an efficient manner [1]. Among these methods, the bidirectional Dijkstra's algorithm with binary heaps computes point-to-point shortest path in $O(|E| + |V| \log |V|)$. Further, the Floyd-Warshall algorithm solves all pairs shortest paths in $O(|V|^3)$. A large number of methods have been designed to improve running time of shortest-path algorithms. For example, contraction hierarchies and arc flags have been successfully used [3].

2.2 Gaussian Process Regression

GPs provide a Bayesian paradigm to learn an implicit functional relationship $y = f(\boldsymbol{x})$ from a training dataset $\{(\boldsymbol{x}_i, y_i); i = 1, 2, ..., n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the vector of observed input variables (i.e., predictors), and y_i is the observed target value. A comprehensive discussion of GPs in machine learning can be found in [13].

GPs directly elicit a prior distribution on the function $f(\boldsymbol{x})$, and assume it to be a GP a priori,

$$f(\boldsymbol{x}) \sim GP\left(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x'})\right). \tag{1}$$

For a new point $\boldsymbol{x}_*$, the goal is to predict $y_* = f(\boldsymbol{x}_*)$. Given that the regression function is a GP, the distribution of the values of f at any finite number of points is a multivariate Gaussian distribution. Therefore,

$$\begin{pmatrix} \boldsymbol{y} \\ y_* \end{pmatrix} \sim \mathcal{N}\left(\mu(x), \begin{pmatrix} K & K'_* \\ K_* & K_{**} \end{pmatrix}\right), \tag{2}$$

where K is the covariance matrix for the labeled points, K_* is the covariance vector between the new point and the labeled points, and K_{**} is the measurement noise. Then,

$$\Pr(y_* \mid \boldsymbol{y}) \sim \mathcal{N}\left(K_* K^{-1} \boldsymbol{y}, K_{**} - K_* K^{-1} K'_*\right). \tag{3}$$

The prediction of a GP model depends on the choice of covariance function, which identifies the expected correlation between the observed data. Typically, a parametric family of functions is used, and the hyperparameters are inferred from the data. Examples of the commonly used covariance functions include polynomial kernel, automatic relevance determination (ARD), and radial basis function (RBF). Methods for learning the hyperparameters are based on maximization of the marginal likelihood, which can be performed using gradient-based optimization algorithms.

3 SYSTEM MODEL

In this section, we present the system model. We first define a model of transportation network. Then, we construct a detector for identifying faulty traffic sensors using a prediction model based on Gaussian Processes.

3.1 Transportation Network

Consider a transportation network modeled as a graph $G = (V, E)$, where edges represent road segments and vertices represent connections between road segments (e.g., traffic junctions). We assume that a subset $S \subseteq E$ of the road segments are monitored by sensors that measure traffic state (e.g., speed, occupancy, flow) at discrete timesteps $k \in \mathbb{N}$. The measurements of these sensors are transmitted to a navigation service, which given a set of queries $Q(k)$ at timestep k, computes the corresponding shortest paths. For segments without a traffic sensor, we assume the navigation service uses either previously computed values or predicted values using measurements of adjacent sensors.

Traffic sensors may be faulty due to miscalibration or hardware failure. If a sensor $s \in S$ is faulty, there is a discrepancy between the actual and measured values. In other words, if $a_s(k)$ is the actual value and $m_s(k)$ is the measured value of faulty sensor s, then $m_s(k) = a_s(k) + \varepsilon_s(k)$, where $\varepsilon_s(k)$ is the fault value at time k. In this model, we do not consider faults that result in no data being sent, since such cases can easily be filtered out by an operator.

3.2 Gaussian Process-Based Detector

Given the sensor measurements, we need to decide whether some sensors are faulty. We assume that the number of sensors that simultaneously become faulty is low, which is true in practice. As a result, for any sensor, the majority of nearby sensors that have not been marked faulty provide reliable traffic data, and so we can use these nearby sensors to predict the value measured by the sensor in question. To detect faults, we then compare the predictions to the measurements, and if there is a significant difference between the predicted values and the received measurements, an alarm indicating presence of a fault in that particular sensor is triggered.

3.2.1 Traffic Prediction. As our traffic predictor, we use GPs, which is a kernel-based machine learning method. Kernel-based methods have gained special attention for traffic prediction because of their generalization capability and superior nonlinear approximation. Among different kernel-based methods, previous work shows that GPs outperform other methods such as ARIMA and neural networks [17]. We use GPs because in addition to the above advantages, it allows for explicit probabilistic interpretation of forecasting outputs.

As the kernel function, we decide for the commonly used ARD squared exponential,

$$K(\boldsymbol{m}(k), \boldsymbol{m}(k)') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{d} \frac{(m_i(k) - m'_i(k))^2}{\sigma_i^2}\right), \tag{4}$$

where $\boldsymbol{m}(k)$ and $\boldsymbol{m}(k)'$ are vectors of measurements, and σ_f and $\{\sigma_i\}_{i=1}^{d}$ are hyperparameters.

We let the target variable be the predicted traffic value p_s (e.g., traffic flow or occupancy) of sensor $s \in S$ at timestep k. Further, we let the predictor variables be the measured traffic values of other sensors at the same timestep. In practice, two sensors are highly correlated if they are in close proximity. Therefore, it is possible to select predictor variables as the measured values of d closest sensors from the target sensor, where the choice of d depends on the network structure. This way, the predicted traffic value is defined as $p_s(k) = f(m_{V(s)}(k))$, where $V(s)$ is the set of d closest sensors from s.

3.2.2 Detection Algorithm. We can efficiently detect failures for each sensor $s \in S$, by comparing the measured traffic value $m_s(k)$ with the predicted traffic value $p_s(k)$. We use Cumulative sum control chart (CUSUM) as the detection algorithm, which is a sequential analysis technique typically used for monitoring change detection [12].

Consider sensor $s \in S$, with a sequence of measurements $m_s(1), ..., m_s(k)$ and corresponding traffic predictions with means $p_s(1), ..., p_s(k)$ and standard deviations $\sigma_s(1), ..., \sigma_s(k)$. The standardized residual signal is defined as

$$z_s(k) = \frac{m_s(k) - p_s(k)}{\sigma_s(k)}. \tag{5}$$

Moreover, upper and lower cumulative sums are defined as,

$$U_s(k) = \max(0, U_s(k-1) + z_s(k) - b_s), \tag{6}$$

$$L_s(k) = \min(0, L_s(k-1) + z_s(k) + b_s), \tag{7}$$

where $U_s(k) = L_s(k) = 0$ for $k = 1$, and b_s is a small constant.

Denoting the detection threshold at timestep k by $\eta_s(k)$, a measurement sequence violates the CUSUM criterion at the sample $z_s(k)$ if it obeys $U_s(k) > \eta_s(k)$ or $L_s(k) < -\eta_s(k)$. Formally, letting H_0 and H_1 be the null and fault hypothesis, the decision rule is described by

$$d_s(U_s(k), L_s(k)) = \begin{cases} H_1 & \text{if } U_s(k) > \eta_s(k) \text{ or } L_s(k) < -\eta_s(k) \\ H_0 & \text{otherwise} \end{cases}. \tag{8}$$

3.2.3 False-Negative and False-Positive Trade-off. In anomaly detectors, there might be a *false negative*, which means failing to raise an alarm when a fault did happen. Further, there might be a *false positive*, which means raising an alarm when the sensor exhibits normal behavior. It is desirable to reduce the FP and FN probabilities as much as possible. But, there exists a trade-off between them, which can be controlled by changing the 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 $FN_s(\eta_s(k))$ is the probability of FN when the threshold is $\eta_s(k)$, given that the sensor is faulty. Similarly, we denote the attainable FP probability for each sensor s by $FP_s : \mathbb{R}_+ \rightarrow [0, 1]$, where $FP_s(\eta_s(k))$ is the FP probability when the threshold is $\eta_s(k)$, given that the sensor is in normal operation. It is possible to plot the FP probability as a function of the FN probability for various threshold values [5] (e.g., see Figure 3).

4 OPTIMAL DETECTION

In this section, we formulate the problem of finding optimal thresholds for anomaly detection of traffic sensors, considering route planning as their primary application. The objective is to select the optimal thresholds for anomaly detectors in order to minimize the losses caused by false positives and false negatives. Then, we present an algorithm to find near-optimal detection thresholds. Finally, we characterize critical sensors, whose failure can have high impact on the traffic application.

4.1 Optimization Problem

First, consider the set of queries Q, and a route planning algorithm that takes as inputs the set of queries and the measured and predicted traffic values, and outputs the optimal routes. For a single query $q \in Q$ and sensor $s \in S$, we denote by $P_q(m_s)$ the optimal route computed using the measured traffic values for all sensors, and we denote by $P_q(p_s)$ the optimal route using the predicted value p_s for sensor s and the measured values $\boldsymbol{m}_{-s}$ for all other sensors. Finally, for a given route r and sensor s, let $T(r, m_s)$ and $T(r, p_s)$ be the total travel time based on the measured m_s and predicted p_s values for sensor s, respectively, and the measured values $\boldsymbol{m}_{-s}$ for all other sensors.

Then, $T(P_q(p_s), m_s)$ is the measured travel time of the shortest route computed using the predicted value p_s for sensor s. Similarly, $T(P_q(m_s), m_s)$ is the measured travel time of the shortest route computed using the measured value m_s. We define the loss caused by a false positive as follows:

$$C_{s,q}^{FP}(p_s, m_s) = T(P_q(p_s), m_s) - T(P_q(m_s), m_s), \tag{9}$$

that is, the difference in measured travel time between using either the predicted or the measured value for sensor s.

The rationale behind the above expression is the following. In case of a FP, according to the detector, the measured value m_s is incorrect, but it is actually correct. Consequently, we choose a route that is computed using our prediction p_s instead of the optimal route, which would be computed using the measurement m_s. To quantify the loss, we need to compare the travel times of the two routes, and we must use the measured traffic value m_s for this comparison since that is the correct value in this case.

Similarly, for a FN, $T(P_q(m_s), p_s)$ is the predicted travel time of the shortest route using measured value m_s, and $T(P_q(p_s), p_s)$ is the predicted travel time of the shortest path using predicted value p_s. The loss caused by a FN is

$$C_{s,q}^{FN}(p_s, m_s) = T(P_q(m_s), p_s) - T(P_q(p_s), p_s), \tag{10}$$

that is, the difference in predicted travel time between using either the measured or the predicted value for sensor s. Note that in (9) and (10), the values of P and T can be computed using existing route planning algorithms [1].

Next, let $FP_s(\eta_s(k))$ and $FN_s(\eta_s(k))$ be the probabilities of false-positive and false-negative errors when detection threshold $\eta_s(k)$ is selected. Further, let p_f be the probability of fault, and let $p_n = 1 - p_f$ be the probability of normal operation. For a given query q, the total loss caused by FPs and FNs is,

$$\begin{aligned} L_{s,q}(\eta_s(k)) = &FP_s(\eta_s(k)) \cdot C_{s,q}^{FP}(p_s, m_s) \cdot p_n + \\ &FN_s(\eta_s(k)) \cdot C_{s,q}^{FN}(p_s, m_s) \cdot p_f \ . \end{aligned} \tag{11}$$

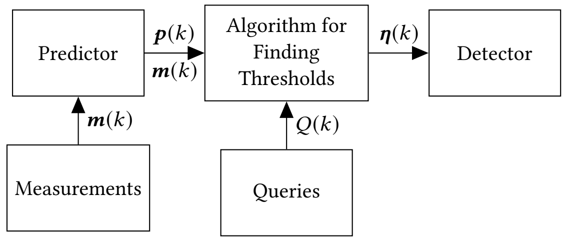


Figure 1: Information flow in our approach.

Considering the set of all queries Q, the total loss is

$$L_s(\eta_s(k), Q) = \sum_{q \in Q} L_{s,q}(\eta_s(k)), \tag{12}$$

which allows us to define the notion of optimal detection threshold for a sensor.

Definition 4.1 (Optimal Detection). The detection threshold $\eta_s^*(k)$ is optimal for sensor s if it minimizes the loss function (12). Formally, $\eta_s^*(k)$ is optimal for sensor s if

$$\eta_s^*(k) \in \operatorname*{argmin}_{\eta_s(k)} L_s(\eta_s(k), Q). \tag{13}$$

Figure 1 shows the flow of information in our approach. At each timestep k, given measurements $\boldsymbol{m}(k)$, the predictor computes the predicted measurements $\boldsymbol{p}(k)$. Then, given a set of queries $Q(k)$, and the predictions and measurements, the thresholds $\boldsymbol{\eta}(k)$ are computed for the detectors using the algorithm presented next.

4.2 Algorithm for Obtaining Thresholds

We present Algorithm 1 to find near-optimal detection thresholds. The algorithm implements a random-restart hill climbing technique. If the FP to FN trade-off curve is convex, which makes (12) convex, we are able to compute optimal thresholds using convex optimization methods. However, this is not generally the case, as trade-off curves tend to be non-convex (see Figure 3 for an instance of a trade-off curve).

The algorithm considers each sensor separately, and finds its corresponding detection threshold. At each iteration, the algorithm selects a new starting point and finds a local minimum using gradient-based optimization. In order to avoid unnecessary computation, we skip computing detection thresholds for sensors with very similar measured and predicted traffic values. Formally, for sensor $s \in E$, we select detection threshold $\eta_s = \infty$, if $|z_s(k)| < b$. This is because the detector's statistics $U_s(k)$ and $L_s(k)$ are decreasing and it is unlikely that an alert would be raised if one was not raised before.

4.3 Critical Sensors

Value of the optimal loss gives insight on the criticality of traffic sensors. Fault on a sensor that has high loss value degrades the system's performance more than fault on a sensor with low loss value. We formally define the set of δ-critical sensors below.

Definition 4.2 (Critical Sensors). Set of δ-critical sensors in a time period $[1, T]$ is defined as the set of sensors which have the average optimal loss values of greater than or equal to δ. That is to say, a sensor s is critical if $\frac{1}{T}\sum_{k=1}^{T} L_s(\eta_s^*(k), Q(k)) \geq \delta$.

Algorithm 1 Algorithm for Obtaining Thresholds

```
1:  **Input** $Q$, $FP(\eta)$, $FN(\eta)$, $\alpha$, $\gamma$
2:  **Initialize:** $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta}_0$, $L^* \leftarrow \infty$
3:  **for all** $s \in S$ **do**
4:      **if** $|z(k)| \leq b$ **then**
5:          $\eta_s^* \leftarrow \infty$
6:      **else**
7:          **while** $i < N$ **do**
8:              $\eta_{s,new} \hookleftarrow FP_s^{-1}(\text{Uniform}([0,1]))$
9:              $\eta_{s,old} \leftarrow 0$
10:             **while** $|L_s(\eta_{s,new}, Q) - L_s(\eta_{s,old}, Q)| > \alpha$ **do**
11:                 $\eta_{s,old} \leftarrow \eta_{s,new}$
12:                 $\eta_{s,new} \leftarrow \eta_{s,old} - \gamma \nabla_{\eta_s} L_s(\eta_{s,old}, Q)$
13:             **if** $L_s(Q, \eta_{s,new}) < L_s^*$ **then**
14:                 $\eta_s^* \leftarrow \eta_{s,new}$
15:                 $L_s^* \leftarrow L_s(\eta_{s,new}, Q)$
16:             $i \leftarrow i + 1$
17: **return** $\boldsymbol{\eta}^*$
```

Identifying critical sensors is beneficial, since it allows us to locate the most vulnerable elements of a network, which should be strengthened first to increase the robustness of a network. For example, if we have a limited budget which permits us to replace only a subset of the sensors with more robust ones, then we should start with the critical sensors.

5 EVALUATION

In this section, we implement our method and evaluate it numerically using a route planning platform.

5.1 System Model

5.1.1 Traffic Data. We use a traffic dataset obtained from the Caltrans Performance Measurement System (PeMS) database [2]. The database provides real-time and historical traffic data from over 39,000 individual sensors, which span the freeway system across metropolitan areas of the State of California. Figure 2 shows the location of sensors in our case study, in which a total of 40 sensors are considered. We use the 5-minute aggregated data collected on the weekdays of September 3, 2016 to September 17, 2016. The dataset contains 115,200 data points. The first 7 days are used as training data, and the remaining 7 days are used as test data.

To simulate faults, we use models for a specific set of fault types and ranges of fault magnitudes, which is similar to the approach presented in [16]. The fault models are: 1) Constant Relative Overcount (caused by e.g., unsuitable sensitivity levels); range: 3% to 7% of the actual values (i.e., $\varepsilon_s(k) = u_s a_s(k)$ where $0.03 \leq u_s \leq 0.07$), 2) Conditional Undercount (caused by e.g., sensor saturation); range: 7% to 13% (i.e., $\varepsilon_s(k) = u_s a_s(k)$ where $-0.13 \leq u_s \leq -0.07$).

Next, for each sensor, we construct a predictor using the measurements of its d closest sensors as the predictor variables. We select $d = 10$ since it results in the minimum overall prediction error. We choose $b_s = 0.05$ for all the detectors, to make them sensitive to small shifts in the mean. We evaluate each detector's performance by plotting the FP probability against the FN probability at various threshold values. Figure 3 shows the trade-off curve of the detector

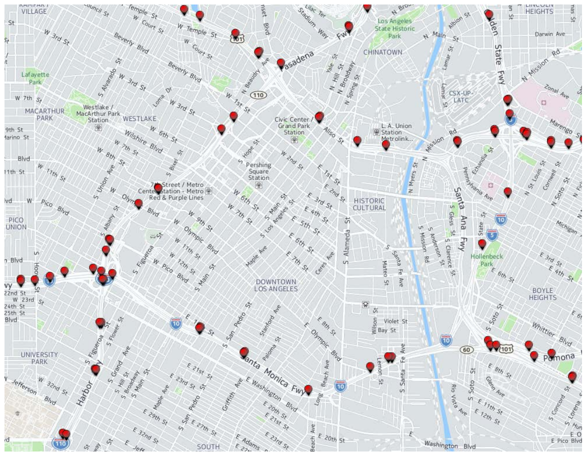


Figure 2: A map of traffic sensors installed in Downtown Los Angeles.

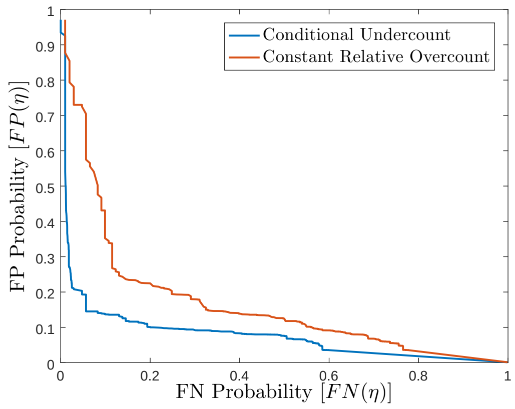


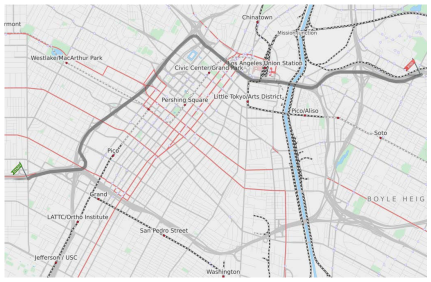
Figure 3: Trade-off between the false-positive and false-negative probabilities.

implemented for a sensor, whose identifier in the PeMS dataset is VDS 774685.

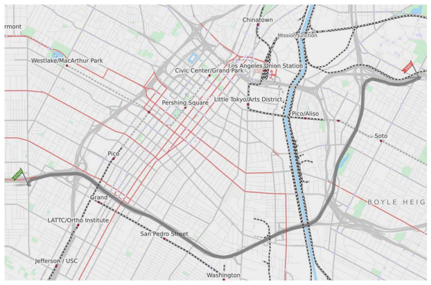
5.1.2 Route Planner. We use OpenTripPlanner (OTP), which is an open source platform for multi-modal route planning [11]. OTP relies on open data standards including OpenStreetMap for street networks. The default routing algorithm in OTP is the A^* algorithm with a cost-heuristic to prune the search. For improved performance on large networks, it also uses contraction hierarchies.

5.2 Results

We simulate a route planning scenario in OTP, where the edge costs (i.e., travel times) are updated using our traffic data. For a source and destination as shown in Figure 4a, we consider 1000 queries made on September 15, from 9:00 am to 10:00 am. Figure 4a shows the shortest route when a particular sensor (i.e., VDS 774685) is healthy, and Figure 4b shows the shortest route when the same sensor has a conditional undercount fault. Note that if the fault remains



(a)



(b)

Figure 4: Reroute occurs due to a conditional undercount fault. (a) Normal. (b) Fault. (Green flag is the source and red flag is the destination.)

undetected (i.e., false negative), a suboptimal route (Figure 4b) will be selected instead of the optimal route (Figure 4a). In another scenario, assume an alarm is triggered under normal operation (i.e., false positive). This means that the predicted value is used for route planning instead of the accurate measurement value, which depending on the prediction accuracy, may result in a suboptimal route planning solution.

We use Algorithm 1 to find near-optimal thresholds that minimize losses due to FPs and FNs. We assume that for each sensor, the probability of fault is $p_f = 0.05$. For the previously considered sensor, at $k = 1$ (i.e., from 9:00 am to 9:05 am), the loss value (12) as a function of the threshold is shown in Figure 5. In this case, Algorithm 1 finds the optimal thresholds. For the Conditional Undercount, the optimal threshold and the minimum loss are $\eta = 0.17$ and $L = 16.2$, whereas for the Constant Relative Overcount, the optimal threshold and the minimum loss are $\eta = 0.39$ and $L = 30.0$.

Further, Table 1 shows the average optimal loss for some sensors, i.e., $\frac{1}{T}\sum_{k=1}^{T} L_s(\eta_s^*(k), Q(k))$. As a baseline, we also compute the minimum loss when the thresholds have static values at all the timesteps. That is, for all k, we assign $\eta_s(k) = \eta_s^*$, where $\eta_s^* \in \text{argmin}_{\eta_s} \sum_k L_s(\eta_s, Q)$. We observe that our method achieves significantly smaller losses compared the static case. The loss values can also be used to identify the set of δ-critical sensors. For example, 50.0-critical sensors are made bold in the table.

6 RELATED WORK

There are many papers that study traffic prediction. The work in [9] uses multivariate kernel regression models to predict traffic flow in

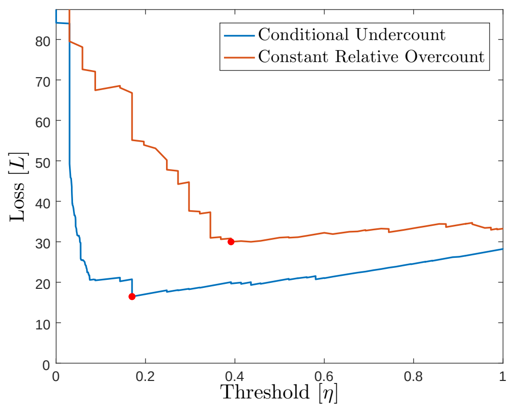


Figure 5: Loss as a function of detection threshold.

Table 1: Average Optimal Losses

Sensor ID	Cond. Undercount		Cons. Rel. Overcount	
	Optimal	Static	Optimal	Static
774685	16.2	31.2	30.0	38.1
774672	18.0	27.6	22.1	36.7
772501	15.6	24.3	12.8	19.2
763453	**51.8**	74.3	**57.5**	80.9
737158	43.0	59.6	**54.8**	71.4

a network, considering route planning as the application. In [4], the paper provides a travel time prediction algorithm in a small scale simulated network. The work in [15] constructs robust algorithms for short-term traffic flow prediction. Finally, in [7], classical time series approaches are used for short-term speed prediction in a network.

The problem of anomaly detection of traffic sensors is reviewed in [10]. The paper categorizes different methods into the three levels of macroscopic, mesoscopic, and microscopic, and provides practical guidelines for anomaly detection. The work in [18] presents three methods to detect faulty traffic measurements. The methods are based on Pearson's correlation, cross-correlation, and multivariate ARIMA. Finally, the work in [14] presents a test, which is based on the relationship between flows at adjacent sensors to detect faulty loop detectors. Nevertheless, since previous papers use static thresholds, their methods result in high losses due to FPs and FNs.

In our previous work, we have considered the problem of optimal parameter selection for anomaly detection. The problem of finding optimal thresholds for intrusion detectors is studied in [8]. 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 [6] 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.

7 CONCLUSIONS

We studied the problem of finding optimal detection parameters for anomaly detection of traffic sensors, considering route planning as application. We constructed a predictor using Gaussian processes, which was then used for anomaly detection. We studied how to find the optimal detection parameters, which minimize losses due to FP and FN errors. We also characterized critical sensors, whose failure can have high impact on the traffic application. We implemented our method and evaluated it numerically using a route-planning platform. Our evaluations indicated that the proposed detection method successfully minimizes the performance losses.

ACKNOWLEDGMENTS

The work is supported by the National Science Foundation (CNS-1238959, CNS-1647015), the Air Force Research Laboratory (FA 8750-14-2-0180), and the National Institute of Standards and Technology (70NANB15H263)

REFERENCES

[1] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. 2016. Route planning in transportation networks. In *Algorithm Engineering*. Springer, 19–80.
[2] Caltrans. 2016. Performance Measurement System (PeMS). Available: http://pems.dot.ca.gov. (2016). [Accessed: 10/25/2016].
[3] Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. 2015. User-constrained multimodal route planning. *Journal of Experimental Algorithmics (JEA)* 19 (2015).
[4] Lili Du, Srinivas Peeta, and Yong Hoon Kim. 2012. An adaptive information fusion model to predict the short-term link travel time distribution in dynamic traffic networks. *Transportation Research Part B: Methodological* 46, 1 (2012), 235–252.
[5] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.
[6] Amin Ghafouri, Waseem Abbas, Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. 2016. Optimal Thresholds for Anomaly-Based Intrusion Detection in Dynamical Environments. In *Proceedings of the Decision and Game Theory for Security: 7th International Conference, GameSec 2016, New York, NY, USA, November 2-4, 2016*, Vol. 9996. Springer, 415.
[7] Yiannis Kamarianakis, Angelos Kanas, and Poulicos Prastacos. 2005. Modeling traffic volatility dynamics in an urban network. *Transportation Research Record: Journal of the Transportation Research Board* (2005), 18–27.
[8] Aron Laszka, Waseem Abbas, S Shankar Sastry, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. 2016. Optimal thresholds for intrusion detection systems. In *Proceedings of the Symposium and Bootcamp on the Science of Security*. ACM.
[9] Thomas Liebig, Nico Piatkowski, Christian Bockermann, and Katharina Morik. 2017. Dynamic route planning with real-time traffic predictions. *Information Systems* 64 (2017), 258–265.
[10] Xiao-Yun Lu, Pravin Varaiya, Roberto Horowitz, and Joe Palen. 2008. Faulty loop data analysis/correction and loop fault detection. In *15th World Congress on Intelligent Transport Systems*.
[11] B McHugh. 2011. The opentripplanner project. *The OpenTripPlanner Project* (2011), 12–16.
[12] ES Page. 1954. Continuous inspection schemes. *Biometrika* 41, 1/2 (1954), 100–115.
[13] Carl Edward Rasmussen. 2006. Gaussian processes for machine learning. (2006).
[14] Stephen Peter Robinson. 2006. *The development and application of an urban link travel time model using data derived from inductive loop detectors*. Ph.D. Dissertation.
[15] Shiliang Sun, Rongqing Huang, and Ya Gao. 2012. Network-scale traffic modeling and forecasting with graphical lasso and neural networks. *Journal of Transportation Engineering* 138, 11 (2012), 1358–1367.
[16] Peter Widhalm, Hannes Koller, and Wolfgang Ponweiser. 2011. Identifying faulty traffic detectors with Floating Car Data. In *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*. IEEE.
[17] Yuanchang Xie, Kaiguang Zhao, Ying Sun, and Dawei Chen. 2010. Gaussian processes for short-term traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board* 2165 (2010).
[18] Nikolas Zygouras, Nikolaos Panagiotou, Ioannis Katakis, Dimitrios Gunopulos, and UOA GR. 2015. Towards detection of faulty traffic sensors in real-time. (2015).