ChangeDAR: Online Localized Change Detection for Sensor Data on a Graph

Bryan Hooi, Leman Akoglu, Dhivya Eswaran, Amritanshu Pandey
Marko Jereminov, Larry Pileggi, Christos Faloutsos
Carnegie Mellon University
Pittsburgh, PA, USA
{bhooi,deswaran,christos}@cs.cmu.edu,{lakoglu,amritanp,mjeremin,pileggi}@andrew.cmu.edu

ABSTRACT

Given electrical sensors placed on the power grid, how can we automatically determine when electrical components (e.g. power lines) fail? Or, given traffic sensors which measure the speed of vehicles passing over them, how can we determine when traffic accidents occur? Both these problems involve detecting change points in a set of sensors on the nodes or edges of a graph. To this end, we propose ChangeDAR (Change Detection And Resolution), which detects changes in an online manner, and reports when and where the change occurred in the graph.

Our contributions are: **1) Algorithm:** we propose novel information theoretic optimization objectives for scoring and detecting localized changes, and propose two algorithms, ChangeDAR-S and ChangeDAR-D respectively, to optimize them. **2) Theoretical Guarantees:** we show that both methods provide constant-factor approximation guarantees (Theorems 5.2 and 6.2). **3) Effectiveness:** in experiments, ChangeDAR detects traffic accidents and power line failures with 75% higher F-measure than comparable baselines. **4) Scalability:** ChangeDAR is online and near-linear in the graph size and the number of time ticks.

CCS CONCEPTS

• **Information systems** → *Data mining*;

KEYWORDS

Online, Localized, Change Detection, Sensor Data, Graph

ACM Reference Format:
Bryan Hooi, Leman Akoglu, Dhivya Eswaran, Amritanshu Pandey Marko Jereminov, Larry Pileggi, Christos Faloutsos. 2018. ChangeDAR: Online Localized Change Detection for Sensor Data on a Graph. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271669

1 INTRODUCTION

How do we detect change points using sensors placed on a subset of the nodes or edges of a graph? In the power grid setting, this question is motivated by the need to quickly detect electrical component failures using sensor data. Such failures can occur due to severe weather, human or equipment failure, or even adversarial intrusion, and have major costs: estimates [4] suggest that reducing outages in the U.S. grid could save $49 billion per year and reduce emissions by 12 to 18%. To achieve this goal, there is a need to use sensor data to quickly identify in real-time when parts of the grid fail, so as to quickly respond to the problem. Similarly, in the traffic setting, a large network of traffic speed detectors spans freeway systems in major metropolitan areas - our goal is to use them to automatically detect notable changes, such as traffic accidents.

In both cases, the changes that we wish to detect are highly **localized**, both in time and with respect to network structure: power line failures affect a localized set of power lines due to redirection of current through neighboring lines, and the same holds for traffic accidents, which slow traffic in neighboring roads.

An additional challenge is that we want to detect change points in an **online** manner: both power grid and traffic data are high-volume and received in real time, since the data comes from sensors which are continuously monitoring. This motivates us to develop fast methods that work in this online setting. When each new data point is received, the algorithm should update itself efficiently - for our algorithm, each update requires constant time, and bounded memory, regardless of the length of the stream.

Thus, our goal is an online, localized change detection algorithm:

Informal Problem 1 (Online Localized Change Detection).

- ***Given*** *a fixed-topology graph $\mathcal{G}$ (e.g. road network or power grid), and time-series sensor values on a subset of the nodes and edges of the graph, received in a streaming manner,*
- ***Find*** *change points incrementally, each consisting of a time t and a localized region of the graph where the change occurred.*

Change detection in time-series [11, 34] and graphs [14, 38] has been studied extensively (expanded in Section 2). Our work differs in two key aspects. Most work focuses on dynamically evolving graphs with changing nodes or edges [14, 38]. In our case the graph topology is fixed, and only sensor values on nodes and edges are evolving. Second, most work in this area finds the time of a change, without being able to localize the change to parts of the graph.

Figure 1 shows an example of our method on traffic data. Given a road graph and the traffic speed over various sensors (indicated by the nodes in the left plots), our method detected a traffic accident.

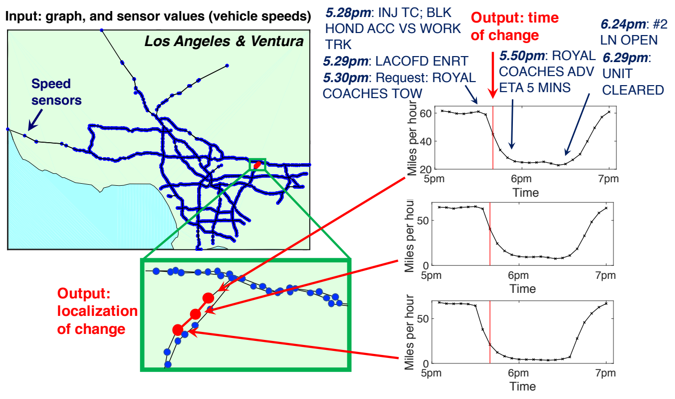


Figure 1: ChangeDAR correctly locates a traffic accident (red): the 3 time-series on the right show drops in average speed at 3 consecutive points on a highway. ChangeDAR outputs the time (red vertical lines) and location (red nodes) of the change, and we verify the accident against the traffic report by the California Highway Patrol (blue text at top-right) [1].

The accident caused a change point (drops in vehicle speed) over a localized set of sensors (red nodes in the left plots). Our method exploits the localized nature of this change to accurately detect it, and outputs both the change time and localization.

Our contributions are as follows:

(1) **Algorithm:** We propose novel information theoretic optimization objectives for 1) scoring and 2) detecting localized changes, and propose two algorithms, ChangeDAR-S and ChangeDAR-D respectively, to optimize them.
(2) **Theoretical Guarantees:** We show that both algorithms provide constant-factor approximation guarantees (Theorems 5.2 and 6.2).
(3) **Effectiveness:** Our algorithms detect traffic accidents and power line failures in a power grid with 75% higher F-measure than comparable baselines in experiments.
(4) **Scalability:** Our full algorithm is online and near-linear in the graph size and the number of time ticks (Figure 6).

Reproducibility: our code and data are publicly available at http://www.andrew.cmu.edu/user/bhooi/changedar.

2 RELATED WORK

Multivariate Change Detection. [5] reviews time-series change detection methods. Multivariate change detection methods aim to segment a time-series into two or more regimes, such as greedy binary segmentation [34], or slower but exact dynamic programming (DP) [20]. PELT [22] applies a pruning step to perform DP in linear time. Other approaches include the Group Fused Lasso (GFL) [7], Bayesian change detection [3], nonparametric methods [25, 27], information-theoretic methods [39], support vector machines [13], and neural networks [26]. These methods do not consider graph structure, and hence do not detect localized changes.

Change Detection in Dynamic Graphs. Change detection methods for dynamic graphs, or graphs which change over time, have been proposed [14, 38]. These include Bayesian methods [31], and distance-based methods, which define a distance metric on graphs: based on diameter [15], node or edge weights [30, 32], connectivity [24], or subgraphs [29]. Except for [30, 32], these do not apply to our setting, as they assume a changing graph over time, while we have a fixed graph with sensor values changing over time.

Graph-based Anomaly Detection and Scan Statistics. A number of methods consider anomaly detection using graph scan statistics [10, 28, 33], which search for a highly anomalous area in static and temporal graphs. [9] uses coloring to efficiently optimize graph scan statistics, and [8] incorporates uncertainty in the observed data. [36] defines the Spatial Scan Statistic while [35] defines the Graph Fourier Scan Statistic (GFSS), which quantify the spatial locality of a signal. These methods focus on time intervals containing unusual activity, while our goal is change detection, which involves a shift between 'regimes' (i.e. generative processes) before and after the change. In the power grid case, our goal is to detect equipment failures, which involve a difference in regimes before and after the failure, but not necessarily a temporal 'burst' of activity at any time.

Table 1 summarizes existing change point detection methods. ChangeDAR differs from existing methods in that it detects localized change points using an online algorithm, and provides theoretical guarantees.

Table 1: Comparison of change detection approaches applicable to sensor data on a graph. The last 2 rows refer to automatically detecting the number of changes, and how many nodes and edges each change affects, respectively.

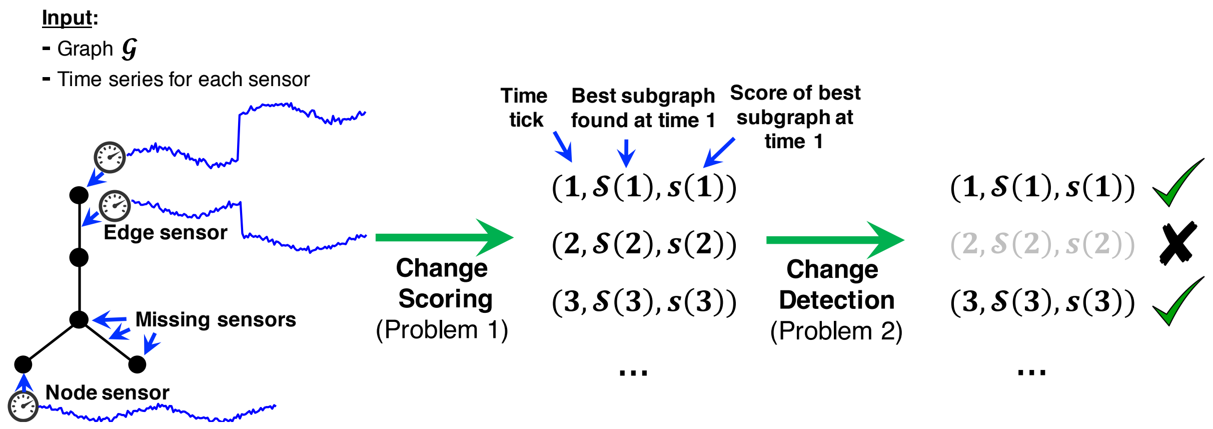
	PELT [22]	GFL [7]	WeightDist [32]	VertexRank [30]	ChangeDAR
Graph-based				✓	✓
Localization					✓
Online Algorithm				✓	✓
Provable Guarantees	✓	✓			✓
Auto Detect # of Changes			✓	✓	✓
Auto Detect Size of Change					✓

3 BACKGROUND

Before delving into the details of our problem statement and proposed algorithms, we briefly review two graph-theoretic concepts that we will make use of later, namely Prize-Collecting Steiner Tree (PCST) and Maximum Weight Independent Set (MWIS).

3.1 Prize-Collecting Steiner Tree (PCST)

The prize-collecting Steiner tree problem [6] finds a connected subgraph of a graph that maximizes total *profit* values on the subgraph nodes while minimizing *cost* of the edges in the subgraph. Intuitively, imagine nodes represent cities, and the *cost* of each edge is the cost of building a road between the two cities, and the *profit* of a node is the profit from that city joining the road network. Then



Input:
- Graph $\mathcal{G}$
- Time series for each sensor
Edge sensor
Missing sensors
Node sensor
Change Scoring (Problem 1)
Time tick
Best subgraph found at time 1
Score of best subgraph at time 1
$(1, \mathcal{S}(1), s(1))$
$(2, \mathcal{S}(2), s(2))$
$(3, \mathcal{S}(3), s(3))$
...
Change Detection (Problem 2)
$(1, \mathcal{S}(1), s(1))$
$(2, \mathcal{S}(2), s(2))$
$(3, \mathcal{S}(3), s(3))$
...

Figure 2: Outline of steps: given a graph with time-series sensors on some nodes and edges, Change Scoring selects the best subgraph where a change occurred at each time, while Change Detection selects the best subset of change points out of these.

PCST aims to construct a road network to maximize net profit (or minimize net cost).

- ***Given*** a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with non-negative node profits $p(v)\ \forall\ v \in \mathcal{V}$ and non-negative edge costs $c(e)\ \forall\ e \in \mathcal{E}$;
- ***Output*** a connected subgraph $\mathcal{S} = (\mathcal{V}_\mathcal{S}, \mathcal{E}_\mathcal{S})$ of $\mathcal{G}$ that minimizes the profit in nodes *not* chosen, plus the total cost of edges chosen, i.e. $\sum_{v \notin \mathcal{V}_\mathcal{S}} p(v) + \sum_{e \in \mathcal{E}_\mathcal{S}} c(e)$.

[21] shows that PCST is NP-hard. [6] introduced the PCST problem and showed a 3-approximation algorithm. [17] proposed a $O(n^2 \log n)$ approximation algorithm with a $2 - \frac{1}{n-1}$ factor guarantee, where n is the number of nodes. [19] improved this to a near-linear time ($O(m \log n)$) 2-approximation algorithm. In our setting, we will use [19] on a modified graph to find localized change points.

3.2 Maximum Weight Independent Set (MWIS)

The maximum weight independent set problem finds a subset of nodes of highest weight which has no edges between them.

- ***Given*** a graph $\mathcal{G}$ with node weights $w(v)\ \forall\ v \in \mathcal{V}$;
- ***Output*** a set $\mathcal{S}$ of nodes with no edges between them maximizing $\sum_{v \in \mathcal{S}} w(v)$.

In the offline setting, approximation algorithms exist [18], but in the online case, [16] showed that no meaningful worst-case guarantees are possible. Hence, follow-up work considers non-worst case analysis [16, 23].

In our setting, we study a constrained variant of the online MWIS problem that arises from our change detection problem, and show that we can obtain constant worst-case approximation guarantees, thanks to certain constraints in our setting. We then use this to obtain theoretical guarantees for multiple change detection.

[9, 33] use PCST for anomaly detection, but to the best of our knowledge, both PCST and online MWIS have not been used for change detection.

4 PROBLEM

4.1 Problem Setting

Table 2 shows the symbols used in this paper.

Table 2: Symbols and definitions

Symbol	Interpretation
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Input graph
n, m	Number of nodes and edges respectively
w	Window size
$X_v(t)$	Sensor value at node v at time t
$X_e(t)$	Sensor value at edge e at time t
k	Number of parameters of time-series model (see Eq. (3))
C_F	Number of bits needed to store a floating point number
$\Delta_v(t)$	'Bitsave' score at node v at time t (see Eq. (1))
$\Delta_e(t)$	'Bitsave' score at edge e at time t (see Eq. (1))
$\pi(v')$	Profit assigned to node v'
$c(e')$	Cost assigned to edge e'
r	Repetitions of randomized algorithm in ChangeDAR-S

We are given an undirected graph $\mathcal{G}$ (e.g. a power grid graph) and a stream of sensor values associated with the nodes and/or edges of the graph (e.g. voltage values measured at nodes), as illustrated in Figure 2. Since some applications involve sensors on nodes while others involve sensors at edges (e.g. current values measured along edges), we consider a general framework that allows for both types of sensors. Some sensors may be missing: hence, if one type of sensor is not present, we can simply consider their values as missing. Define $X_v(t)$ as the sensor value at node v at time t, and $X_e(t)$ as the sensor value at edge e at time t.

We consider the sensor values to arrive in an **online** manner. However, intuitively, it is unrealistic to decide if a change point exists at time t given only information up to time t, since we have no information about what comes after. Hence, we allow for a **window** of w time ticks, in which the algorithm has access to the next w time ticks before needing to make a decision at time t. In practice, this means the algorithm reports whether a change occurred at time t after a lag time of w time ticks.

Hence, given data up to time $t + w$, we would like the algorithm to output if a change occurred at time t. This can be broken down into two sub-problems: 1) change scoring and 2) change detection.

Problem 1 (Online Change Scoring).

- ***Given*** *a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a stream of (possibly missing) sensor values $X_v(t)$ and $X_e(t)$ for each node v, each edge e, and time tick $t = 1, 2, \cdots$:*
- ***Output*** *(in an online manner) at time $t + w$:*
  - ***Scoring****: a score $s(t)$ measuring our confidence that a change occurred at time t;*
  - ***Localization****: the best subset of nodes and edges $\mathcal{S} = \mathcal{V}_{\mathcal{S}} \cup \mathcal{E}_{\mathcal{S}}$, where $\mathcal{V}_{\mathcal{S}} \subseteq \mathcal{V}$ and $\mathcal{E}_{\mathcal{S}} \subseteq \mathcal{E}$, where a change occurred.*

The notion of the 'best' subset where a change most likely occurred, and the interpretation of the score $s(t)$, will be formalized in an information theoretic manner in Section 5.1.

Given these scores, the next sub-problem is to decide which of these changes actually occurred:

Problem 2 (Online Change Detection).

- ***Given*** *a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a stream of sensor values $X_v(t)$ and $X_e(t)$ for each node v, each edge e, and time tick $t = 1, 2, \cdots$:*
- ***Output*** *(in an online manner) at time $t + w$: whether a change occurred at time t, and if so, the corresponding subset $\mathcal{S}(t)$ where the change occurred.*

Figure 2 provides an outline of our approach. We consider Change Scoring in Section 5 and Change Detection in Section 6.

5 CHANGE SCORING: CHANGEDAR-S

We now propose an algorithm for Problem 1, for finding the best subset $\mathcal{S}$ and score $s(t)$ at time t, in an online manner.

5.1 Optimization Objective

We first define our '*Bitsave*' metric, which intuitively measures how beneficial (in terms of number of bits we save) it is to add a change point at time t.

Description Length Framework. First consider the simpler question of how to encode a single time-series $Y_1, \cdots, Y_n$. The Minimum Description Length (MDL) approach states that given data Y, we should encode it using the model M that minimizes the **description length** of Y, which is defined as $\text{Cost}(Y) = \text{Cost}(M) + \text{Cost}(Y|M)$, where $\text{Cost}(M)$ is the number of bits needed to encode the model M, and $\text{Cost}(Y|M)$ is the number of bits needed to encode the data given model M. The full expression for these costs depends on the type of model used, which we describe as follows.

Model Cost: We use a flexible approach that allows for any model family to be used for encoding a time-series, e.g. Autoregression, Seasonal Autoregression etc., resulting in a vector of fitted values $(\hat{Y}_i)_{i=1}^n$. Let k be the number of parameters used: e.g. if we fit an Autoregressive $AR(p)$ model, then $k = p + 1$ (the additional 1 is for the intercept). Then the model cost $\text{Cost}(M)$ is the cost of k floating point values, or $k \cdot C_F$, where C_F is the cost to encode a floating point number[1]. A simple default choice (which we use in our experiments) is to set $\hat{Y}_i$ as the mean $\frac{1}{n}\sum_{j=1}^n Y_j$ for all i, which has $k = 1$ parameter, but more complex functions can be used, particularly when seasonality is present.

[1]We use $C_F = 32$ for standard 4-byte floats.

Data Cost: For the data cost $\text{Cost}(Y|M)$, we assume that the errors follow a normal $\mathcal{N}(0, \sigma^2)$ distribution. The log probability density of the errors $Y_i - \hat{Y}_i$ under this distribution is

$$\log P(Y_1 - \hat{Y}_1, \cdots, Y_n - \hat{Y}_n) = \log \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_i - \hat{Y}_i)^2}{2\sigma^2}} \quad (1)$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^n (Y_i - \hat{Y}_i)^2 + \text{const.} \quad (2)$$

where 'const.' does not depend on the data. By the theory on Huffman coding [12], the cost in bits to represent some data under a given distribution is the negative log-likelihood of the data. Intuitively, the less probable a particular outcome is, the more bits we need to encode it: e.g. encoding the outcome of a coin flip requires 1 bit, but encoding that we rolled a '1' on a fair 8-sided die needs $-\log_2 \frac{1}{8} = 3$ bits.

Taking the negative of (2), the data cost $\text{Cost}(Y|M)$ is $\frac{1}{2\sigma^2}\sum_{i=1}^n (Y_i - \hat{Y}_i)^2$. Thus the total cost (in bits) of encoding Y is:

$$\begin{aligned}\text{Cost}(Y_1, \cdots, Y_n) &= \text{Cost}(M) + \text{Cost}(Y_1, \cdots, Y_n|M) \\ &= k \cdot C_F + \frac{1}{2\sigma^2}\sum_{i=1}^n (Y_i - \hat{Y}_i)^2\end{aligned} \quad (3)$$

Note that we are not actually making a strong requirement that the errors be normal: Eq. (3) simply encodes the data such that the lower the total squared error, the fewer bits we need. We are essentially minimizing squared error, with an additional penalty for the cost of the model, in bits.

Computing Bitsave Scores. We now apply Eq. (3) to the sensor at node v at time t. Since we are in an online setting, bits saved have to be computed with respect to a window of size w. Intuitively, the bits saved from adding a change at time t is the cost of encoding X_v from time $t - w$ to $t + w$, minus the same cost if we were to add a change at time t:

Definition 1 (Bitsave). *The bitsave score from adding a change at time t at node v is:*

$$\begin{aligned}\Delta_v(t) &= \mathit{Cost}(X_v(t-w), \cdots, X_v(t+w)) \\ &- \mathit{Cost}(X_v(t-w), \cdots, X_v(t-1)) \\ &- \mathit{Cost}(X_v(t), \cdots, X_v(t+w))\end{aligned} \quad (4)$$

The bitsave score for edge e, $\Delta_e(t)$, are the same, except using X_e instead of X_v. For missing sensors, we simply set their bitsave score to 0 for all t.

Lemma 1. *A lower bound for $\Delta_v(t)$ (or $\Delta_e(t)$) is:*

$$\Delta_v(t) \geq -k \cdot C_F \quad (5)$$

Proof. When adding a change point at time t, the total data cost cannot increase, since we are allowed to fit the same model to both sides of the change point. Meanwhile, the model cost increases by exactly $k \cdot C_F$, since adding one change point adds an additional set of model parameters, which costs $k \cdot C_F$ bits. □

Localized Change Scoring Objective. The notion of *localized* changes captures the fact that in many applications, changes tend to spread along the graph: e.g. power lines going down affect a connected set of lines, and traffic congestion affects a connected set of roads. To capture this, we stipulate that $\mathcal{S}$, the subgraph of nodes and edges affected by the change, must be connected in the graph.

Hence, at time t, we want to detect a localized set of nodes and edges which is a good change point, i.e. provides large total bits saved. Denote by $\mathcal{S} = \mathcal{V}_\mathcal{S} \cup \mathcal{E}_\mathcal{S}$ the set of nodes and edges we are searching for, where $\mathcal{V}_\mathcal{S} \subseteq \mathcal{V}$ and $\mathcal{E}_\mathcal{S} \subseteq \mathcal{E}$.

Given subgraph $\mathcal{S}$, the *total bitsave* by adding a change point at time t is the sum of Δ values on $\mathcal{S}$: i.e. $\sum_{v \in \mathcal{V}_\mathcal{S}} \Delta_v(t) + \sum_{e \in \mathcal{E}_\mathcal{S}} \Delta_e(t)$. Under the MDL framework, we then need to encode $\mathcal{S}$ itself. The simplest way is to encode each of its elements individually: $\mathcal{S} \subseteq \mathcal{V} \cup \mathcal{E}$ is a subset over $m+n$ elements, so each element takes $\log(m+n)$ bits, or $|\mathcal{S}| \log(m+n)$ bits in total. In summary, the optimization objective to find $\mathcal{S}$ is:

$$\max_{\mathcal{S} \subseteq \mathcal{V}} \; f_t(\mathcal{S}) = \sum_{v \in \mathcal{V}_\mathcal{S}} \Delta_v(t) + \sum_{e \in \mathcal{E}_\mathcal{S}} \Delta_e(t) - |\mathcal{S}| \log(m+n) \tag{6}$$

subject to: $\mathcal{S}$ is connected

5.2 Optimization Approach

To solve (6) we rewrite it into an equivalent form that can be optimized using the Prize-Collecting Steiner Tree framework, then solve it in near-linear time with an approximation guarantee of 2.

We first construct a new, bipartite graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$: it has a node v' for each $v \in \mathcal{V}$, and a node e' for each $e \in \mathcal{E}$. Then, in $\mathcal{G}'$, connect v' to e' iff v is adjacent to e in the original graph G.

For each $v \in \mathcal{V}$, assign the node v' a *profit* of $\pi(v') = \Delta_v(t) + k \cdot C_F$. Similarly, for each $e \in \mathcal{E}$, assign the node e' a *profit* of $\pi(e') = \Delta_e(t) + k \cdot C_F$. Finally, assign each edge $e' \in \mathcal{E}'$ a *cost* of $c(e') = k \cdot C_F + \log(m+n)$. Intuitively, the *profits* correspond to bits saved due to the corresponding sensor, while the *costs* correspond to a model complexity penalty in bits due to encoding each additional node ($\log(m+n)$), as well as the additional set of model parameters due to the added change point ($k \cdot C_F$).

We will show that our optimization objective (6) is equivalent to the Prize-Collecting Steiner Tree objective in the new graph $\mathcal{G}'$.

$$\min_{\mathcal{S}' \subseteq \mathcal{V}'} \; f'_t(\mathcal{S}') = \sum_{v' \notin \mathcal{V}_{\mathcal{S}'}} \pi(v') + \sum_{e' \in \mathcal{E}_{\mathcal{S}'}} c(e') \tag{7}$$

subject to: $\mathcal{S}'$ is connected

Let $\mathcal{S} \subseteq \mathcal{V} \cup \mathcal{E}$, and define the corresponding $\mathcal{S}'$ to include all nodes in $\mathcal{G}'$ corresponding to nodes and edges in $\mathcal{S}$, i.e. $\mathcal{S}' = \{v' : v \in \mathcal{V}_\mathcal{S}\} \cup \{e' : e \in \mathcal{E}_\mathcal{S}\}$.

LEMMA 2. *The objectives of* (6) *and* (7) *are equivalent:*

$$f'_t(\mathcal{S}') = -f_t(\mathcal{S}) + C \tag{8}$$

where $C = \sum_{v' \in \mathcal{V}'} \pi(v') - k \cdot C_F - \log(m+n)$ *is constant w.r.t.* $\mathcal{S}$ *and* $\mathcal{S}'$.

PROOF. $\mathcal{S}'$ has 1 node for each node or edge of $\mathcal{S}$, so it has $|\mathcal{S}|$ nodes. Moreover, any optimal $\mathcal{S}'$ must be a tree since any edge in a cycle in $\mathcal{S}'$ only increases costs without adding any profits. Hence, $\mathcal{S}'$ has $|\mathcal{S}| - 1$ edges. Then:

$$\begin{aligned}
f'_t(\mathcal{S}') &= \sum_{v' \notin \mathcal{V}_{\mathcal{S}'}} \pi(v') + \sum_{e' \in \mathcal{E}_{\mathcal{S}'}} c(e') \\
&= \sum_{v' \notin \mathcal{V}'} \pi(v') - \sum_{v' \in \mathcal{V}_{\mathcal{S}'}} \pi(v') + (|\mathcal{S}'| - 1)(\log(m+n) + k \cdot C_F) \\
&= C - \sum_{v' \in \mathcal{V}_{\mathcal{S}'}} \pi(v') + |\mathcal{S}|(\log(m+n) + k \cdot C_F) \\
&= C - (\sum_{v \in \mathcal{V}_\mathcal{S}} \Delta_v(t) + \sum_{e \in \mathcal{E}_\mathcal{S}} \Delta_e(t) - |\mathcal{S}| \log(m+n)) \\
&= C - f_t(\mathcal{S}).
\end{aligned}$$

□

Moreover, $\mathcal{S}'$ is connected if and only if its nodes form a single connected component, which is equivalent to $\mathcal{S}$ being connected. In combination with Lemma 2, this implies that minimizing $f'_t(\mathcal{S}')$ is equivalent to maximizing $f_t(\mathcal{S})$.

Finally, by Lemma 1, we have $\Delta_v(t) \geq -k \cdot C_F$ and $\Delta_e(t) \geq -k \cdot C_F$, which implies that $\pi(v') \geq 0 \; \forall \; v' \in \mathcal{V}_{\mathcal{S}'}$, so all profits are nonnegative. The costs, $\log(m+n) + k \cdot C_F$ are also nonnegative. Hence we can solve (7) in near-linear time with approximation guarantees of 2 using algorithms for Prize-Collecting Steiner Tree (as reviewed in Section 3.1).

Algorithm 1 gives the full CHANGEDAR-S algorithm. We first convert $\mathcal{G}$ to the bipartite $\mathcal{G}'$ (Line 1). Then for each t, we compute the PCST profits (Lines 4 to 5) and costs (Line 6). We then solve the resulting PCST problem in $\mathcal{G}'$ (Line 8) to obtain the best subgraph $\mathcal{S}'$, which we then convert back to a subgraph of $\mathcal{G}$ (Line 10). Finally, the score $s(t)$ is the total bitsave of $\mathcal{S}$, defined as $f_t(\mathcal{S})$ in (6).

Algorithm 1: CHANGEDAR-S change scoring algorithm

```
Input  : Graph $\mathcal{G}$, sensor stream $X_v(t), X_e(t)$ over time,
         window size $w$
Output : For each $t$, the best change-point localized set $\mathcal{S}(t)$,
         and its bitsave score $s(t)$
1  Convert $\mathcal{G}$ to $\mathcal{G}'$ s.t.: $(v', e') \in \mathcal{E}'$ iff $v$ is adjacent to $e$ in $\mathcal{G}$
2  while sensor values for time tick $t + w$ are received do
3      ▷Compute profits $\pi$ and costs $c$
4      $\pi(v') = \Delta_t(v) + k \cdot C_F$
5      $\pi(e') = \Delta_t(e) + k \cdot C_F$
6      $c(\cdot) = k \cdot C_F + \log(m+n)$
7      ▷Solve (7) using Prize-Collecting Steiner Tree
8      $\mathcal{S}' = \text{PCST}(\mathcal{G}', \pi(\cdot), c(\cdot))$
9      ▷$\mathcal{S}$ is the nodes and edges corresponding to $\mathcal{S}'$
10     $\mathcal{S}(t) = \{v : v' \in \mathcal{S}'\} \cup \{e : e' \in \mathcal{S}'\}$
11     ▷$s(t)$ is the total bitsave of $\mathcal{S}$
12     $s(t) = f_t(\mathcal{S})$
```

5.3 Theoretical Results

THEOREM 5.1. *Algorithm 1 is online, requiring bounded memory and linear time.*

Proof. **Memory.** At time t, the only data we need to store over time are the sensor values from time $t-w$ to $t+w$ for $O(m+n)$ sensors, used for computing (1), while takes $O(w(m+n))$ memory, which is bounded regardless of the stream length.

Running time. Computing all $m+n$ bitsave scores at time t takes $O(C(m+n))$ time, where C is the time to fit $\hat{Y}$ in Eq. (2), which depends on the model family being used. For AR, seasonal AR and the constant (mean) model, C is (amortized) constant, independent of w, as we show in our supplementary document [2]. Then Lines 4 to 12 are linear in $m+n$, while the PCST step takes $O((m+n)\log n)$ time. Hence, Algorithm 1 is online, requiring $O((m+n)(C+\log n))$ per time step, or $O(T(m+n)(C+\log n))$ for T time ticks. □

Theorem 5.2. *Algorithm 1 provides an approximation guarantee of* 2 *for optimizing* (7)*: letting $\mathcal{S}'$ be the returned set and $\mathcal{S}^*$ the optimal solution:*

$$f'_t(\mathcal{S}') \leq 2 \cdot f'_t(\mathcal{S}^*). \tag{9}$$

Proof. The optimization objective (7) is in the Prize-Collecting Steiner Tree form. By [19], this can be solved in near-linear time with an approximation guarantee of 2. □

6 CHANGE DETECTION: CHANGEDAR-D

So far, for each time tick $t = 1, 2, \cdots$, we have found a localized change at subgraph $\mathcal{S}(t)$, with score $s(t)$. Our goal now is to decide which of them are actually change points: i.e. to output, in an online manner, a set of time ticks $\{t_1, t_2, \cdots\} \subseteq \{1, 2, \cdots\}$ and corresponding subsets $\mathcal{S}(t_1), \mathcal{S}(t_2), \cdots$ such that a change occurred in each subset $\mathcal{S}(t_i)$ at time t_i.

6.1 Optimization Objective

Intuitively, since the scores $s(t)$ represent bits saved, we want to pick the best set of time ticks $\{t_1, t_2, \cdots\} \subseteq \{1, 2, \cdots\}$ maximizing total bits saved, but without selecting conflicting change points. Recall that a change point $(t, \mathcal{S}(t))$ is scored based on its bits saved in the sensors in $\mathcal{S}(t)$ from time $t-w$ to $t+w$. This means that for two change points t_i and t_j, if their subgraphs $\mathcal{S}(t_i)$ and $\mathcal{S}(t_j)$ have nonempty intersection, and their time intervals also overlap (i.e. $|t_i - t_j| \leq 2w$), then these two change points **conflict**, i.e. they cannot both be chosen in the final set of changes. In this way we define the **conflict graph**:

Definition 2 (Conflict Graph). *The* **conflict graph** $\mathcal{G}_{conf} = (\mathcal{V}_{conf}, \mathcal{E}_{conf})$ *has a node for each time tick:* $\{1, 2, \cdots\}$. *For $t_i, t_j \in \{1, 2, \cdots\}$, it has an edge between t_i and t_j iff the two change points $(t_i, \mathcal{S}(t_i))$ and $(t_j, \mathcal{S}(t_j))$ conflict, i.e.:*

$$|t_i - t_j| \leq 2w, \quad \textit{and} \quad |\mathcal{S}(t_i) \cap \mathcal{S}(t_j)| > 0.$$

Our goal is to choose the set of non-conflicting change points with highest total bitsave. As an optimization objective:

$$\begin{aligned} &\max_{\mathcal{T} \subseteq \{1,2,\cdots\}} \; g(\mathcal{T}) = \sum_{t \in \mathcal{T}} s(t) \\ &\textbf{subject to: } (t_i, t_j) \notin \mathcal{E}_{\text{conf}} \; \forall \; (t_i, t_j) \in \mathcal{T} \end{aligned} \tag{10}$$

6.2 Optimization Approach

Objective (10) is equivalent to a Maximum Weight Independent Set (MWIS) problem, where we put weights of $s(t)$ on node t, and find the max weight set which is independent (i.e. has no edges) in graph $\mathcal{G}_{\text{conf}}$. The key challenge, however, is that we need to optimize (10) in an **online** manner in which we receive nodes in $\mathcal{G}_{\text{conf}}$, along with their weight and conflicting edges, incrementally. As reviewed in Section 3.2, [16] showed that no meaningful worst-case guarantees are possible for the general online MWIS problem.

In our case, however, we have a **constrained** online MWIS problem: each node can only conflict with the $2w$ nodes to its immediate past and future. This turns out to be sufficient for solving the problem with constant approximation guarantee (treating w as fixed).

We optimize (10) using a hybrid greedy-randomized approach, which keeps track of 1 greedy solution, and r randomized solutions, and returns the best solution out of these at each time. The greedy solution performs better in practice, but the randomized algorithm provides better theoretical guarantees. Hence, performing both algorithms and returning whichever gives a higher objective value gives the 'best of both worlds' in terms of both empirical performance and theoretical guarantees.

The full algorithm is given in Algorithm 2. Lines 7 to 13 perform a greedy choice: if $s(t)$ exceeds the score of all its conflicting neighbors (Line 8), we add t into the greedy set $\mathcal{T}^{(0)}$ (Line 10) and remove all its conflicting neighbors (Line 11). Based on the changes we made to $\mathcal{T}^{(0)}$, we then update $g_{\mathcal{T}^{(0)}}$ (a variable keeping track of $g(\mathcal{T}^{(0)})$ i.e. Objective (10)) accordingly (Line 13).

Lines 14 to 22 perform a randomized choice: we sample a uniform random value u_t from 0 to 1 for each time t (Line 16). In the ith repetition, if u_t is greater than the random values of all its neighbors, we add t to the set $\mathcal{T}^{(i)}$ and remove its neighbors (Lines 19 to 20). As before we then update $g_{\mathcal{T}^{(i)}}$ accordingly (Line 22).

6.3 Theoretical Results

Theorem 6.1. *Algorithm 2 is online, requiring bounded memory and linear time.*

Proof. **Memory.** At time t, we only need to store the nodes and edges of $\mathcal{G}_{\text{conf}}$ for up to the past $2w$ time ticks, since all neighbors of the node at time t are at time $t-2w$ or later. This requires $O(w)$ for the nodes and at most $O(w^2)$ for the edges. For the randomized part, we need to store r uniform values for each of these nodes, requiring $O(wr)$ memory. Thus the total memory usage is $O(w(r+w))$. We show that a constant value of r is sufficient in Theorem 6.2.

Running time. Each iteration of the greedy part takes $O(w)$ to visit each neighbor, and $O(w)$ in the same way for each of the r repetitions of the randomized part. Hence the overall running time is $O(wr)$ per iteration, or $O(wrT)$ overall for T time ticks. □

Theorem 6.2. *Algorithm 2 has constant-factor approximation guarantee in expectation: letting $\mathcal{T}_{best}$ be the output of Algorithm 2 and $\mathcal{T}^*$ be the optimal solution of* (7)*:*

$$\mathbb{E}[g(\mathcal{T}_{best})] \geq \frac{1}{4w+1} g(\mathcal{T}^*) \tag{11}$$

Proof. Consider a fixed repetition i of the randomized algorithm, at any current time tick t'. For each node t, t is included in

Algorithm 2: ChangeDAR-D change detection algorithm

```
Input  : Conflict graph $\mathcal{G}_{\text{conf}}$, window size $w$, and change sets
         with scores $(t, \mathcal{S}(t), s(t))$ as a stream at time
         $t = 1, 2, \cdots$ from ChangeDAR-S
Output : Set of change points
         $\mathcal{T}_{\text{best}} = \{(t_1, \mathcal{S}(t_1)), (t_2, \mathcal{S}(t_2)), \cdots\}$
1  ▷Change sets $\mathcal{T}^{(i)}$, and corresponding objective values $g_{\mathcal{T}^{(i)}}$
2  $\mathcal{T}^{(i)} = \{\} \ \forall\ i = 0, 1, \cdots, r$
3  $g_{\mathcal{T}^{(i)}} = 0 \ \forall\ i = 0, 1, \cdots, r$
4  while node for time tick $t + w$ is received do
5      $\mathcal{N}_t = \{t' : (t, t') \in \mathcal{E}_{\text{conf}}\}$
6      ▷Run Greedy and Randomized and choose the best
       obtained set:
7      ▷ Greedy : if $t$'s score exceeds sum of neighbors' scores:
8      if $s(t) > \sum_{t' \in \mathcal{N}_t} s(t')$ then
9          ▷add $t$ into $\mathcal{T}^{(0)}$ and remove its neighbors
10         $\mathcal{T}^{(0)} = \mathcal{T}^{(0)} \cup \{t\}$
11         $\mathcal{T}^{(0)} = \mathcal{T}^{(0)} \setminus \mathcal{N}_t$
12         ▷update objective for $\mathcal{T}^{(0)}$ based on changes to it
13         $g_{\mathcal{T}^{(0)}} = g_{\mathcal{T}^{(0)}} + s(t) - \sum_{t' \in \mathcal{N}_t} s(t')$
14     ▷ Randomized : over $r$ repetitions:
15     for $i$ in 1 to $r$ do
16         Sample $u_t^{(i)} \sim$ Uniform(0, 1)
17         ▷If $t$'s value is greater than its neighbors' values
18         if $u_t^{(i)} > \max_{t' \in \mathcal{N}_t} u_{t'}^{(i)}$ then
19             $\mathcal{T}^{(i)} = \mathcal{T}^{(i)} \cup \{t\}$
20             $\mathcal{T}^{(i)} = \mathcal{T}^{(i)} \setminus \mathcal{N}_t$
21             ▷update objective for $\mathcal{T}^{(i)}$ based on changes to it
22             $g_{\mathcal{T}^{(i)}} = g_{\mathcal{T}^{(i)}} + s(t) - \sum_{t' \in \mathcal{N}_t} s(t')$
23     Output $\mathcal{T}_{\text{best}} = \arg\max_{\mathcal{T} \in \{\mathcal{T}^{(0)}, \cdots, \mathcal{T}^{(r)}\}} g_{\mathcal{T}}$
```

$\mathcal{T}^{(i)}$ iff $u_t^{(i)}$ is greater than its neighbors' values. Since t can only be neighbors with the time ticks from $t - 2w$ to $t + 2w$, it has at most $4w$ neighbors. Thus, $u_t^{(i)}$ is greater than all its neighbors with probability $\geq \frac{1}{4w+1}$.

We next compute the expected value of $g(\mathcal{T}^{(i)})$. In the following, (12) is the definition of g, (13) is by linearity of expectation, (14) is since t is included with probability $\geq \frac{1}{4w+1}$, and (15) is since $g(\mathcal{T}^*)$ is a sum of some subset of the $s(\cdot)$ score values.

$$\mathbb{E}[g(\mathcal{T}^{(i)})] = \mathbb{E}\left[\sum_{t \in \mathcal{T}^{(i)}} s(t)\right] \tag{12}$$

$$= \sum_{t=1}^{t'} s(t) \cdot P(t \in \mathcal{T}^{(i)}) \tag{13}$$

$$\geq \sum_{t=1}^{t'} \frac{s(t)}{4w+1} \tag{14}$$

$$\geq \frac{1}{4w+1} g(\mathcal{T}^*) \tag{15}$$

□

The guarantee in Theorem 6.2 is a guarantee in expectation, but can also be converted to a 'with high-probability' guarantee:

Theorem 6.3. *Algorithm 2 has constant-factor approximation guarantee with high probability: formally, for any* $0 < \delta, \varepsilon < 1$, *as long as we set* $r \geq \frac{\log \delta}{\log(\frac{4w}{4w+\varepsilon})}$, *then with probability at least* $1 - \delta$:

$$g(\mathcal{T}_{best}) \geq \frac{1-\varepsilon}{4w+1} g(\mathcal{T}^*) \tag{16}$$

Proof. For any i, note that $g(\mathcal{T}^{(i)}) \leq g(\mathcal{T}^*)$, and by Theorem 6.2, $\mathbb{E}[\frac{g(\mathcal{T}^{(i)})}{g(\mathcal{T}^*)}] \geq \frac{1}{4w+1}$. By Markov inequality on the nonnegative variable $1 - \frac{g(\mathcal{T}^{(i)})}{g(\mathcal{T}^*)}$:

$$P(\frac{g(\mathcal{T}^{(i)})}{g(\mathcal{T}^*)} \leq \frac{1-\varepsilon}{4w+1}) = P(1 - \frac{g(\mathcal{T}^{(i)})}{g(\mathcal{T}^*)} \geq \frac{4w+\varepsilon}{4w+1}) \tag{17}$$

$$\leq \frac{1 - \mathbb{E}[\frac{g(\mathcal{T}^{(i)})}{p} g(\mathcal{T}^*)]}{\frac{4w+\varepsilon}{4w+1}} \tag{18}$$

$$\leq \frac{1 - \frac{1}{4w+1}}{\frac{4w+\varepsilon}{4w+1}} = \frac{4w}{4w+\varepsilon} \tag{19}$$

Since $g(\mathcal{T}_{\text{best}})$ takes the max of r independent repetitions:

$$P(g(\mathcal{T}_{\text{best}}) \leq \frac{1-\varepsilon}{4w+1}) = (\frac{4w}{4w+\varepsilon})^r \tag{20}$$

$$\leq \delta \text{ if } r \geq \frac{\log \delta}{\log(\frac{4w}{4w+\varepsilon})} \tag{21}$$

□

For example, set ε and δ to small constants, e.g. 0.01. Then this theorem implies that $P(g(\mathcal{T}_{\text{best}}) \geq \frac{0.99}{4w+1})$ holds with probability at least 0.99 as long as r is at least a constant value.

7 EXPERIMENTS

We design experiments to answer the following questions:

- **Q1. Change Detection Accuracy:** how accurate are the change times detected by ChangeDAR?
- **Q2. Localization Accuracy:** how accurate are the change locations reported by ChangeDAR?
- **Q3. Scalability:** how does our method scale with data size?

Our code (in Matlab) and data are publicly available at http://www.andrew.cmu.edu/user/bhooi/changedar. Experiments were done on a 2.4 GHz Intel Core i5 Macbook Pro, 16 GB RAM running OS X 10.11.2. For window size w, small values around 5 are a good default, as larger values might miss short-lasting changes. Hence we use $w = 5$. We set error variance σ^2 (Eq. (3)) to 0.05 for the power grid data and 1.5 for the traffic data.

Data. We use 4 power grid graphs of different sizes, `PolandHVN1`, `PolandHVN2`, `PolandHVN3`, and `PolandHVN4`, which are real power grids from different parts of the Polish high voltage network [40], as well as a traffic dataset, `TrafficLA` [1]. In the power grids, nodes represent electrical buses, edges represent transmission lines, and sensors have hourly frequency. `TrafficLA` [1] consists of the road network of the 'Los Angeles and Ventura' district in California, in the first week of January 2018. The nodes are traffic sensors that record the average speed passing through that point within each

5 minute interval. The edges of the graph form the road network. The data also contains traffic incident reports by the California Highway Patrol, which report the latitude, longitude, time, duration and description of incidents (e.g. traffic accidents). Dataset details are in Table 3.

Table 3: Datasets used.

Dataset name	Nodes	Edges	Time Ticks	Domain	Sensors
`PolandHVN1` [40]	2383	2896	480	Power	Voltage
`PolandHVN2` [40]	2737	3506	480	Power	Voltage
`PolandHVN3` [40]	3012	3572	480	Power	Voltage
`PolandHVN4` [40]	3120	3693	480	Power	Voltage
`TrafficLA` [1]	4828	4868	2016	Traffic	Speed

7.1 Q1. Detection Accuracy

In this section, we compare ChangeDAR against baseline change detection approaches, in their accuracy for detecting power line failures simulated using Matpower [40], a standard power system simulation program.

Experimental Settings. For each graph, out of 480 time ticks (hourly data for 20 days) we sample 10 random time ticks as the times when changes occur. In each such time tick, we deactivate a randomly chosen edge (i.e. no current can flow over that edge) for the remainder of the time period. As input to Matpower, we use load patterns estimated from real data [37] from the Carnegie Mellon University (CMU) campus for 20 days from July 29 to August 17, 2016, with its standard deviation scaled down by a factor of 10.

Given this input, each algorithm returns a set of time ticks where it detected a change. We evaluate this using F-measure ($\frac{2\cdot\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$) compared to the true set of anomalies.

Baselines. We compare ChangeDAR to the following change detection methods:

- *Dynamic Graph Change Detection:* WeightDist [32] and VertexRank [30].
- *Multivariate Change Detection:* PELT [22] and Group Fused Lasso (GFL) [7].
- *Graph-based Scan Statistics:* Graph Fourier Scan Statistic (GFSS) [35].

Only our method and VertexRank are online; for VertexRank, the original algorithm actually requires an offline standard deviation, but we assume this can be done online as well. We use the offline version of VertexRank in our experiments.

For WeightDist, we select the ARMA orders using AIC, with an upper limit of 2 following the original paper [32]. For VertexRank we use a threshold of 2 standard deviations, following the original paper [30]. GFL and GFSS require the number of changes as input: we pass in the true number of changes. For GFL we use the default LARS (Least Angle Regression) approach. For GFSS, we preprocess by converting the time series to adjacent differences over time, and set ρ as the 5th-percentile smallest eigenvalue, following the original paper [35].

Results. Figure 3 shows an example of a power failure correctly detected, shown by the blue cross. This change causes cascading effects in the voltage levels in the surrounding nodes, which ChangeDAR is able to detect (red circles). We evaluate the set

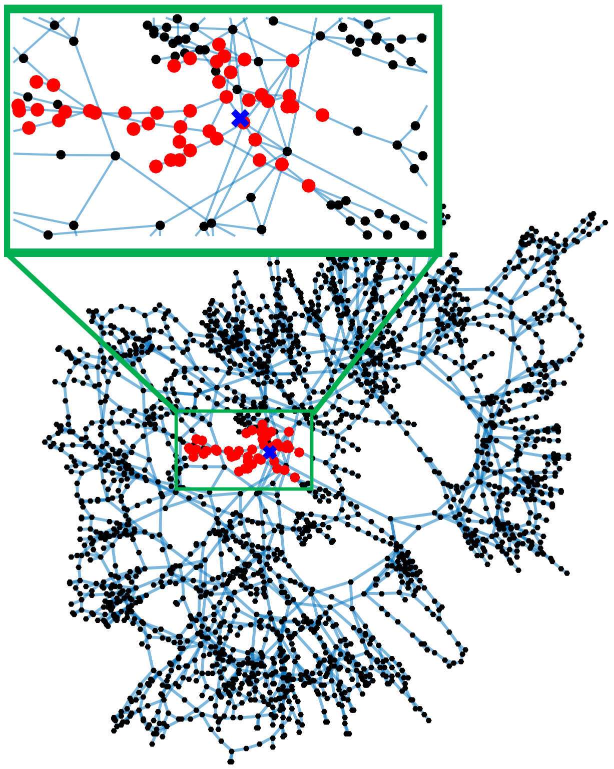


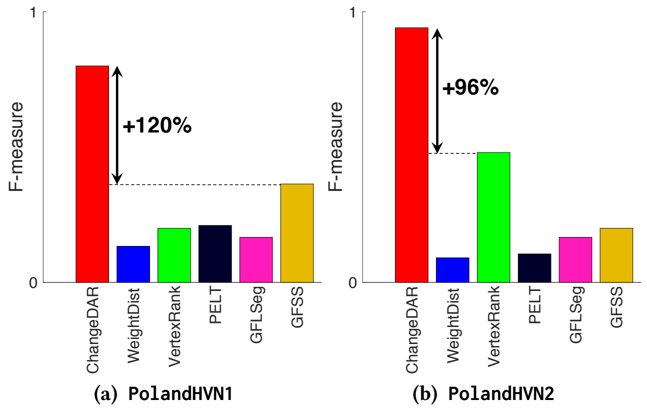
Figure 3: ChangeDAR correctly detects a power failure: the blue cross shows a power line failure, causing cascading effects in surrounding voltage levels, which are detected as a localized change-point by ChangeDAR (red circles).

of change times output by each method against the ground truth values using F-measure on the 4 datasets in Figure 4a to 4d. The results show that ChangeDAR outperforms the baselines by 75% or more. Since only ChangeDAR detects changes that are localized in the graph and persist over a period of time, this suggests that combining graph and temporal structure in this way is effective. Note that this occurs despite the baselines (other than VertexRank) being offline algorithms, while our method is online.

7.2 Q2. Localization Accuracy

We evaluate ChangeDAR in detecting and locating traffic accidents in the `TrafficLA` data. Figure 1 shows a traffic incident reported in the incident report as a traffic collision at 5.30pm on Jan 3 2018. ChangeDAR reported a localized change at the time shown by the red vertical line and at the three stations marked in red, all of which experienced sharp drops in average speed shortly after the accident.

We now evaluate the accuracy of ChangeDAR against the ground truth traffic accidents, compared to the same set of baselines (with



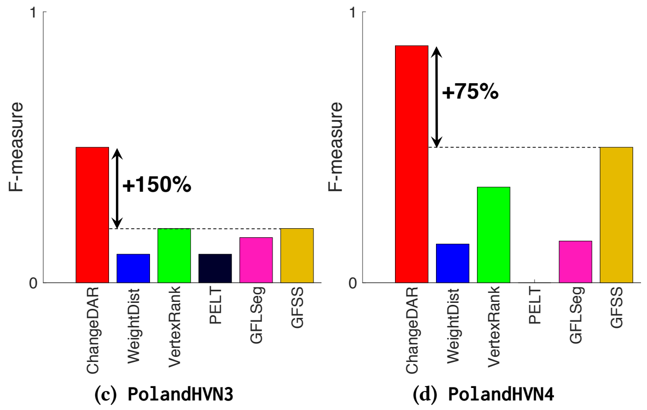


Figure 4: CHANGEDAR accurately detects power line failures: F-measure of detecting transmission line failures compared to (mostly offline) baselines.

the same settings as before). Since none of the baselines perform localization (i.e. returning the location of an anomaly), for each change point returned by a baseline, we select the sensor with the largest negative change in speed at that time as the detected location. We only consider negative changes since traffic accidents should only result in decreases in vehicle speed.

The ground truth accidents come from incident reports by the California Highway Patrol. Each incident is accompanied by its occurrence time and location. We use all events listed as traffic collisions with duration at least 1 hour. For each change reported by an algorithm, we match it to a ground truth incident if the two occurred at most 1 hour apart, and the mean location among the algorithm's detected nodes is at most '*radius*' away from the true location, for the values of '*radius*' plotted on the x-axis of Figure 5.

Figure 5 shows that CHANGEDAR outperforms the same baselines in precision and recall of locating traffic accidents, by 70% and 227% respectively, and F-measure by 124%. The fairly low precision and recall of all methods occurs because many traffic accidents do not lead to any discernible change in vehicle speed, and conversely, traffic slowdowns may be caused by regular congestion or events that are not reported as traffic accidents.

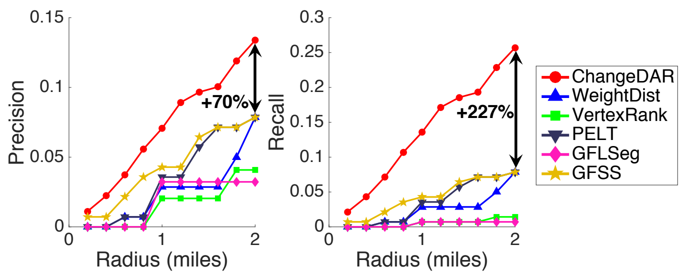


Figure 5: CHANGEDAR accurately locates traffic accidents: CHANGEDAR outperforms baselines in precision (left) and recall (right) on ground truth traffic accidents. The x-axis plots the radius used to determine whether a change point output by each algorithm matches a ground truth accident.

7.3 Q3. Scalability

Finally, we verify that CHANGEDAR scales linearly in the number of time ticks and the number of edges in the graph. Figure 6a plots the wall-clock time taken for CHANGEDAR to run on the TrafficLA dataset, varying the number of time ticks from 10%, 20%, $\cdots$, 100% of the full number of time ticks. For Figure 6b we construct subsets of nodes by taking the 10%, 20%, $\cdots$, 100% of nodes with lowest geographical latitude, and run CHANGEDAR on the induced subgraphs of these subsets. Subsetting via latitude is done to prevent the graph from separating into a large number of connected components. Figures 6a and 6b show that CHANGEDAR scales linearly in both dimensions.

CHANGEDAR is fast, taking 10ms on average to run on each time tick on our largest (TrafficLA) graph, with 4828 sensor values per time tick.

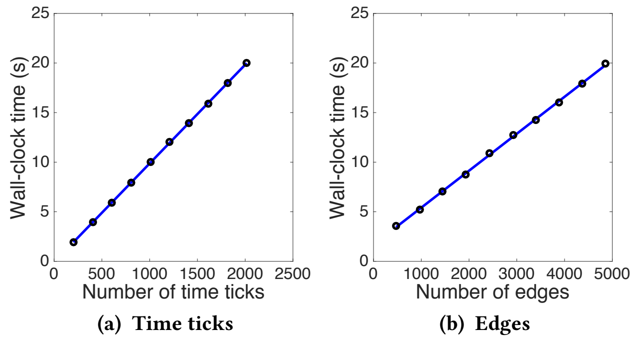


Figure 6: CHANGEDAR scales linearly: wall-clock time of CHANGEDAR against (a) number of time ticks and (b) number of edges.

8 CONCLUSION

In this paper, we propose online algorithms for detecting localized changes for sensor data on a graph. This type of data occurs in many settings: e.g. power grid monitoring, traffic, climate, disease surveillance, and monitoring users on social networks. Our approach uses

PCST and online MWIS, which to the best of our knowledge, have not been used for change detection. Our contributions are:

(1) **Algorithm:** We propose novel information theoretic optimization objectives for 1) scoring and 2) detecting localized changes, and propose two algorithms, ChangeDAR-S and ChangeDAR-D respectively, to optimize them.
(2) **Theoretical Guarantees:** We show that both algorithms provide constant-factor approximation guarantees (Theorems 5.2 and 6.2).
(3) **Effectiveness:** Our algorithms detect traffic accidents and power line failures in a power grid with 75% or more higher F-measure than comparable baselines in experiments.
(4) **Scalability:** Our full algorithm is online and near-linear in the graph size and the number of time ticks (Figure 6).

9 ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1314632, IIS-1408924, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, and in part by the Defense Advanced Research Projects Agency (DARPA) under award no. FA8750-17-1-0059 for the RADICS program. 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 National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] 2018. Caltrans Performance Measurement System. http://pems.dot.ca.gov/. (2018). Accessed: 2018-04-08.

[2] 2018. Supplementary Document. www.dropbox.com/sh/bg2n1uknw16o2as/AABAg5HoMxtjQ4PcwroZuBISa?dl=0. (2018).

[3] Ryan Prescott Adams and David JC MacKay. 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742* (2007).

[4] S Massoud Amin. 2011. US grid gets less reliable [The Data]. *IEEE Spectrum* 48, 1 (2011), 80–80.

[5] Samaneh Aminikhanghahi and Diane J Cook. 2017. A survey of methods for time series change point detection. *Knowledge and information systems* 51, 2 (2017), 339–367.

[6] Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson. 1993. A note on the prize collecting traveling salesman problem. *Mathematical programming* 59, 1-3 (1993), 413–420.

[7] Kevin Bleakley and Jean-Philippe Vert. 2011. The group fused lasso for multiple change-point detection. *arXiv preprint arXiv:1106.4199* (2011).

[8] Jose Cadena, Arinjoy Basak, Anil Vullikanti, and Xinwei Deng. 2018. Graph Scan Statistics With Uncertainty. (2018).

[9] Jose Cadena, Feng Chen, and Anil Vullikanti. 2017. Near-Optimal and Practical Algorithms for Graph Scan Statistics. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 624–632.

[10] Feng Chen and Daniel B Neill. 2014. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1166–1175.

[11] Haeran Cho and Piotr Fryzlewicz. 2015. Multiple-change-point detection for high dimensional time series via sparsified binary segmentation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 77, 2 (2015), 475–507.

[12] Thomas M Cover and Joy A Thomas. 2012. *Elements of information theory*. John Wiley & Sons.

[13] Frédéric Desobry, Manuel Davy, and Christian Doncarli. 2005. An online kernel change detection algorithm. *IEEE Transactions on Signal Processing* 53, 8 (2005), 2961–2974.

[14] Dongsheng Duan, Yuhua Li, Yanan Jin, and Zhengding Lu. 2009. Community mining on dynamic weighted directed graphs. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*. ACM, 11–18.

[15] Matthew E Gaston, Miro Kraetzl, and Walter D Wallis. 2006. Using graph diameter for change detection in dynamic networks. *Australasian Journal of Combinatorics* 35 (2006), 299.

[16] Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. 2014. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *International Colloquium on Automata, Languages, and Programming*. Springer, 508–519.

[17] Michel X Goemans and David P Williamson. 1995. A general approximation technique for constrained forest problems. *SIAM J. Comput.* 24, 2 (1995), 296–317.

[18] Magnús M Halldórsson. 2004. Approximations of weighted independent set and hereditary subset problems. In *Graph Algorithms And Applications 2*. World Scientific, 3–18.

[19] Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt. 2015. A nearly-linear time framework for graph-structured sparsity. In *International Conference on Machine Learning*. 928–937.

[20] Brad Jackson, Jeffrey D Scargle, David Barnes, Sundararajan Arabhi, Alina Alt, Peter Gioumousis, Elyus Gwin, Paungkaew Sangtrakulcharoen, Linda Tan, and Tun Tao Tsai. 2005. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters* 12, 2 (2005), 105–108.

[21] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.

[22] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. 2012. Optimal detection of changepoints with a linear computational cost. *J. Amer. Statist. Assoc.* 107, 500 (2012), 1590–1598.

[23] Nitish Korula and Martin Pál. 2009. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages, and Programming*. Springer, 508–520.

[24] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. 2013. Deltacon: A principled massive-graph similarity function. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 162–170.

[25] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. 2013. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks* 43 (2013), 72–83.

[26] X Liu and RG Lathrop Jr. 2002. Urban change detection based on an artificial neural network. *International Journal of Remote Sensing* 23, 12 (2002), 2513–2518.

[27] David S Matteson and Nicholas A James. 2014. A nonparametric approach for multiple change point analysis of multivariate data. *J. Amer. Statist. Assoc.* 109, 505 (2014), 334–345.

[28] Misael Mongiovi, Petko Bogdanov, Razvan Ranca, Evangelos E Papalexakis, Christos Faloutsos, and Ambuj K Singh. 2013. Netspot: Spotting significant anomalous regions on dynamic networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 28–36.

[29] Misael Mongiovi, Petko Bogdanov, and Ambuj K Singh. 2013. Mining evolving network processes. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 537–546.

[30] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. 2010. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications* 1, 1 (2010), 19–30.

[31] Leto Peel and Aaron Clauset. 2015. Detecting Change Points in the Large-Scale Structure of Evolving Networks.. In *AAAI*. 2914–2920.

[32] Brandon Pincombe. 2005. Anomaly detection in time series of graphs using arma processes. *Asor Bulletin* 24, 4 (2005), 2.

[33] Polina Rozenshtein, Aris Anagnostopoulos, Aristides Gionis, and Nikolaj Tatti. 2014. Event detection in activity networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1176–1185.

[34] Andrew Jhon Scott and M Knott. 1974. A cluster analysis method for grouping means in the analysis of variance. *Biometrics* (1974), 507–512.

[35] James Sharpnack, Alessandro Rinaldo, and Aarti Singh. 2016. Detecting anomalous activity on networks with the graph Fourier scan statistic. *IEEE Transactions on Signal Processing* 64, 2 (2016), 364–379.

[36] James Sharpnack, Aarti Singh, and Alessandro Rinaldo. 2013. Changepoint detection over graphs with the spectral scan statistic. In *Artificial Intelligence and Statistics*. 545–553.

[37] Hyun Ah Song, Bryan Hooi, Marko Jereminov, Amritanshu Pandey, Larry Pileggi, and Christos Faloutsos. 2017. PowerCast: Mining and forecasting power grid sequences. In *ECML-PKDD*. Springer, 606–621.

[38] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. 2007. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 687–696.

[39] Kenji Yamanishi and Kohei Miyaguchi. 2016. Detecting gradual changes from data stream using MDL-change statistics. In *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 156–163.

[40] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. 2011. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems* 26, 1 (2011), 12–19.