Anomaly Detection in Graphs and Time Series: Algorithms and Applications

Bryan Hooi

School of Computer Science and Dietrich College of Humanities and Social Sciences
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: anomaly detection, unsupervised, graph, time series, dynamic graphs, fraud detection, tensor

Abstract

With the increasing availablility of web-scale graphs and high-frequency sensor data, anomaly detection in massive datasets has seen growing focus. Social networks such as Facebook and Twitter contain up to billions of users. Similarly, large scale sensor data includes networks of traffic speed detectors that span freeway systems in major metropolitan areas, networks of voltage sensors spanning the electrical power grid, as well as numerous types of industrial, weather and environmental sensors. These datasets have created an increasing need for scalable algorithms that can automatically analyze this data and flag users or events which are anomalous or of interest.

This thesis focuses on these problems, by developing scalable, principled algorithms that detect unusual behavior or events. Our approaches focus on the use of connectivity and temporal information. These approaches are useful for large graphs such as social networks, as well as for sensor datasets such as traffic data, which contain multiple sensors varying over time, that are also arranged on a graph.

First, we focus on static graphs, in which only connectivity information is present. For example, how can we detect fraudsters in a user-product review graph, or a Twitter follower-followee graph? We first propose a probabilistic approach that evaluates how surprising a dense subgraph is, based on empirical patterns about how dense subgraphs of various sizes typically are. We then consider how to detect dense subgraphs in a way that prevents anomalous users from being able to avoid detection by manipulating their features. Our FRAUDAR approach [HSB$^+$16b] improving detection accuracy by up to 70% F-measure over comparable baselines, and detects a Twitter subgraph of more than 4000 accounts, a majority of which used follower-buying services.

Next, we consider time series: for example, how can we detect anomalous events, such as electrical component failures in power grid time series? We develop algorithms for modelling and detecting anomalies in two main settings: ratings data, in which a set of users rate a set of products; and power grid data, in which modelling and anomaly detection is most accurately done with the use of physics-based circuit models. Our STREAMCAST approach [HSP$^+$18] approach provides an online approach for forecasting and detecting anomalies in time series data, with 27% lower forecasting error than baselines on real data.

Finally, we consider dynamic graphs, where we have graph-structured data over time. In particular, we consider a set of sensors arranged in a graph, each of which collects data over time: for example, traffic speed sensors, which are arranged on a road network, or voltage sensors, which are arranged on a power grid network. We develop algorithms for detecting critical or anomalous events, such as traffic accidents or power line failures. In addition, our GRIDWATCH [HES$^+$18] approach studies how to near-optimally select locations for new sensors to be placed on a power grid graph, improving the detection of component failures by 59% or more F-measure.

Contents

Chapter 1

Introduction

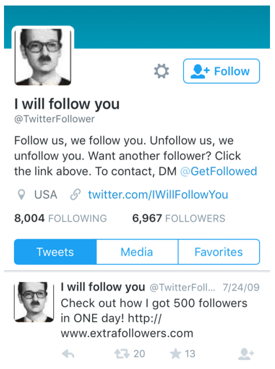
A capacity for surprise is an essential aspect of our mental life, and surprise itself is the most sensitive indication of how we understand our world and what we expect from it.

Daniel Kahnemann, *Thinking, Fast and Slow*

Our brains contain unconscious but sophisticated machinery for detecting deviations from normality and responding to them quickly. In *Thinking, Fast and Slow*, Kahnemann writes: “Studies of brain responses have shown that violations of normality are detected with astonishing speed and subtlety. In a recent experiment, people heard the sentence ‘Earth revolves around the trouble every year.’ A distinctive pattern was detected in brain activity, starting within two-tenths of a second of the onset of the odd word. Even more remarkable, the same brain response occurs at the same speed when a male voice says, ‘I believe I am pregnant because I feel sick every morning’ ... A vast amount of world knowledge must instantly be brought to bear for the incongruity to be recognized [KE11].”

A similar challenge is faced in software systems. In many applications, extremely high-volume data arrives constantly, and the challenge for algorithms is to perform the role that attention does in our brains: of rapidly recognizing deviations from normality in order to surface and respond to the most pressing anomalies as quickly as possible. Hence, with the increasing availablility of web-scale graphs and high-frequency sensor data, anomaly detection in massive datasets has seen growing focus. Social networks such as Facebook and Twitter contain up to billions of users. Similarly, large-scale sensor data involves numerous types of weather and environmental sensors, networks of voltage sensors spanning the electrical power grid, and networks of traffic speed detectors that span freeway systems in major metropolitan areas.

These datasets are creating an increasing need for scalable algorithms that can automatically analyze this data and flag users or events which are anomalous or of interest. For example, in online commerce and social networks, we are often interested in automatically flagging potentially fraudulent users, spam, network intrusions or other forms of attacks. In large-scale sensor data, we are often interested in unusual events or deviations from normal behavior, such as traffic accidents, major weather events, or the breakdown of electrical components in power systems data.



I will follow you
@TwitterFollower

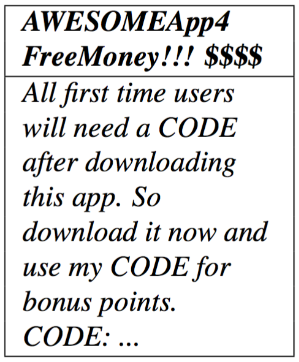
Follow us, we follow you. Unfollow us, we unfollow you. Want another follower? Click the link above. To contact, DM @GetFollowed

USA twitter.com/IWillFollowYou

8,004 FOLLOWING 6,967 FOLLOWERS

I will follow you @TwitterFoll... 7/24/09
Check out how I got 500 followers in ONE day! http:// www.extrafollowers.com

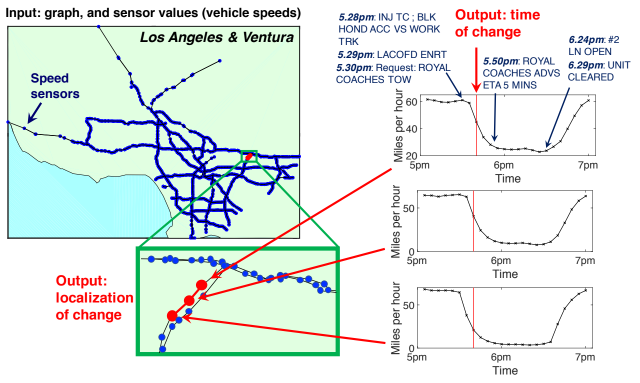
(a) Fake followers



AWESOMEApp4 FreeMoney!!! $$$$

All first time users will need a CODE after downloading this app. So download it now and use my CODE for bonus points. CODE: ...

(b) Fake reviews



(c) Traffic accidents

Figure 1.1: Applications of our anomaly detection approaches. (a) **Social networks:** we detect a large group of fake followers in Twitter. (b) **Online reviews:** we detect fake reviews in an online application market. (c) **Traffic:** we detect ground truth traffic accidents in a dataset containing traffic speed sensors on a road network.

Figure 1.1 shows examples of anomalies in different domains detected by our approaches. In Figure 1.1a, we use connectivity information to detect a large group of fake followers in Twitter. In Figure 1.1b, using temporal information, we detect fake reviews such as this review which advertises 'codes' instead of reviewing the product: we detect these using the temporally bursty nature of groups of fake reviews. In Figure 1.1c, we combine connectivity and temporal information to detect traffic accidents based on traffic speed data: traffic accidents affect a small localized set of nodes on the road traffic graph, and cause a drop in traffic speed over a short time frame. In this figure, nodes represent speed sensors. Red nodes indicate the 3 nodes detected by our algorithm. The 3 plots on the right show the drops in traffic speed on these sensors, which was detected by our method as a change point, and the dark blue text shows ground truth traffic reports, which indicate that a traffic accident occurred.

This thesis focuses on developing techniques that are useful for large graphs such as social networks, as well as sensor datasets, which often involve multiple sensors arranged on a graph and varying over time. In particular, we aim to develop methods for making use of both connectivity and temporal information:

- **Q1. Graphs:** How can we identify subgraphs with unusual connectivity in a graph?
- **Q2. Time Series:** How can we use temporal information to detect time periods containing unusual activity?
- **Q3. Dynamic Graphs:** Given time-varying sensor data arranged on a graph, how can we detect the time and location of unusual events?

1.1 Overview and Contributions

- **Graphs:** We develop a probabilistic score for how abnormal a subgraph is, and an efficient algorithm for optimizing this score [HSLF18]. We then develop an algorithm for detecting dense subgraphs under a setting of 'camouflage,' where the adversaries are able to add edges (such as Twitter follows) in order to evade detection, and show that our algorithm detects fraudsters effectively and with theoretical guarantees in this setting [HSB+16b].
- **Time Series:** Time series data arising from online systems often involves discrete events (such as likes, follows, page views etc.) accompanied by timestamps, while the more traditional time series sensor setting involves real-valued time series data. For the former, we develop an anomaly detection algorithm for ratings data (i.e. ratings submitted by users for products) in order to catch fraudsters attempting to manipulate the ratings for a product [HSB+16a]. For the latter, we develop a forecasting and anomaly detection algorithm for power systems data that exploits domain knowledge via a physics-based model of power systems data [HSP+18]. We also develop an online nonparametric anomaly detection approach that can be used on categorical, numeric or ordinal data [HF18].
- **Dynamic Graphs:** Given a set of sensors arranged on a graph, such as traffic sensors arranged on a road network, we develop algorithms for detecting change points (e.g. traffic accidents) that occurred at a certain time which affects a connected subgraph of sensors [HAE+18]. We then consider the case where we need to select locations to place the sensors to have the highest probability of detecting an anomaly, and propose an algorithm for near-optimally selecting locations for sensor placement [HES+18].

Chapter 2

Graphs

How can we identify subgraphs with unusual connectivity in a static graph?

Static graphs, which contain only connectivity information, are commonly used in practice when temporal information is not present. In this chapter, we consider how to detect groups of nodes with unusual connectivity, such as network intrusion attacks, and link spam in social networks. In many practical settings, real graphs are very sparse, so the groups of nodes that stand out as being abnormal are the sufficiently **dense subgraphs**, i.e. groups of nodes with high connectivity. Hence, TellTail focuses on providing a probabilistic score for how abnormal a subgraph is, while Fraudar provides an algorithm for detecting dense subgraphs that is robust against adversarial manipulation.

2.1 TellTail: Scoring Dense Subgraphs

Section based on work that is under review [HSLF18].

Motivation Given a phone call graph, or a graph of following relationships on Twitter, how can we design a principled measure for identifying surprisingly dense subgraphs? If 50 users each reviewed almost all the same 500 products several times each, can we measure our confidence that this indicates anomalous or fraudulent behavior, while exonerating the users if this is within normal variation? Dense subgraphs often indicate interesting structure, such as network attacks in network traffic graphs, or protein families in protein interaction networks. However, most existing dense subgraph measures, such as average degree, do not take normal variation into account and hence cannot exonerate normal subgraphs, and can also be biased when comparing subgraphs of different sizes. Hence, we aim to propose principled probabilistic measures which solve these problems.

Problem 2.1: Scoring Dense Subgraphs

- **Given:** graph $G = (V, E)$, and a subgraph induced by $S \subseteq V$
- **Output:** $f(S)$, an estimate of how surprising the edge count of S is compared to the distribution of subgraphs of the same size.

2.1.1 Main Ideas

To compare subgraphs of different sizes in a fair manner, we **control for size** by estimating the distribution of edge counts (or 'mass') of subgraphs of each size, and evaluating each subgraph with respect to the distribution of subgraphs of the same size. Then, since our measure is then in 'units' of probability, it can be compared across sizes in a principled way.

A natural question then is: how do we probabilistically model the distribution of subgraph masses of each size? [JBC$^+$16] assumes that the graph follows an Erdos-Renyi distribution, and therefore models the distribution of edge counts using a Poisson distribution. However, the Erdos-Renyi model is unrealistic, and ignores the clustering effects present in real graphs. Indeed, we find empirically that in real graphs, much denser subgraphs exist than expected under Erdos-Renyi models.

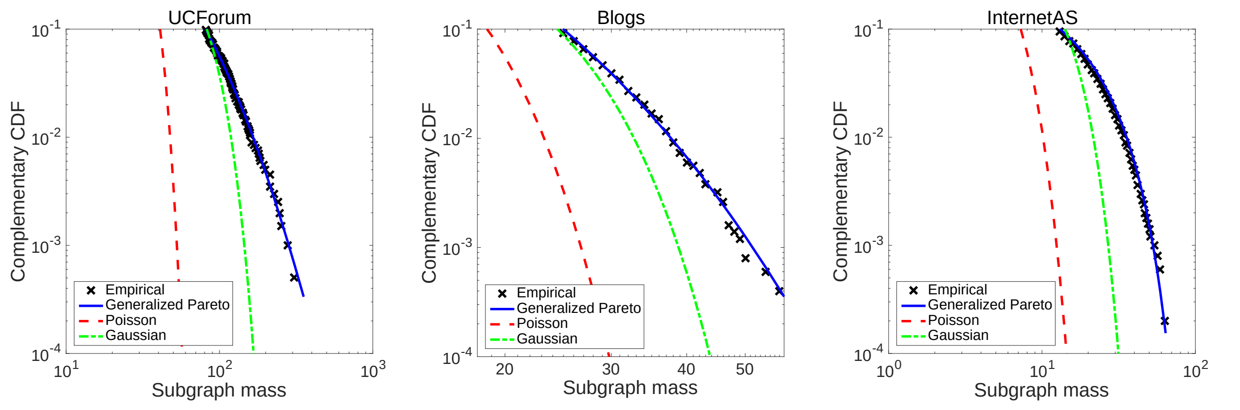
A key idea is that in practice, the large majority of subgraphs are very sparse: however, the subgraphs we are actually interested are those in the 'upper tail' of this distribution, i.e. the dense subgraphs. This suggests the use of approaches from extreme value theory, in particular the Generalized Pareto (GP) Distribution, which is designed for modelling extreme tails of a distribution. The 'universality' property, or Pickands-Balkema-de Haan theorem [PI75], intuitively states that the GP distribution can approximate the tails of any distribution with high accuracy.

Empirical Observations Empirically, we show that GP Distribution fits the empirical distribution of subgraph edge counts of real graphs much better than other standard distributions:

Figure 2.1 shows the empirical distribution of subgraph masses in three real graphs. The crosses show the empirical CCDF (i.e. $1-$ CDF) of the mass of 5000 random subgraphs of size $k = \lfloor \sqrt{n} \rfloor$. The 3 colored lines are maximum likelihood fits of 3 distributions to these masses. The wide gap between the Poisson curve and the crosses indicates that the Poisson distribution greatly underestimates how many dense subgraphs we should observe. Figure 2.1 shows that Gaussian distributions also decay too quickly, while the GP distribution fits the empirical distribution very closely.

Observation 2.1

The upper tail of the subgraph mass distributions of real graphs closely follows a GP distribution.



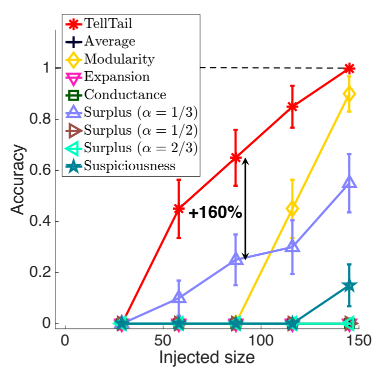
(a) `UCForum` dataset ($n = 899$) (b) `Blogs` dataset ($n = 1.2$K) (c) `InternetAS` dataset ($n = 34$K)

Figure 2.1: **The GP distribution closely fits mass distributions of real graphs:** Black crosses indicate the empirical distribution of subgraph masses for subgraphs of size $k = \lfloor\sqrt{n}\rfloor$, in the form of its complementary CDF (i.e. $1 -$ CDF). The colored curves are the best fit GP, Poisson, and Gaussian to the empirical distribution.

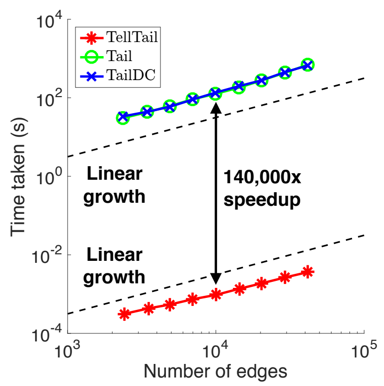
Using a number of additional empirical observations, we propose the TellTail measure, which probabilistically evaluates the surprisingness of a subgraph with respect to a GP distribution. TellTail can be computed in linear time. In addition, we propose a pruning-based algorithm for heuristically optimizing TellTail, and show a guarantee that the pruning will never eliminate nodes from the optimal subset.

2.1.2 Results

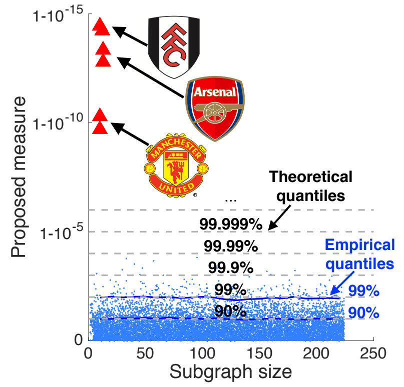
Figure 2.2a shows that this approach as used in our TellTail measure outperforms existing measures in the accuracy of identifying injected subgraphs. In our full results, TellTail outperforms baselines by 70% or more F-measure over 8 datasets. Figure 2.2b shows that our algorithms run in linear time; also, TellTail uses power-law patterns that we observe in real graphs to achieve large speedups of *140,000* times over our more naive sampling-based approach (as used in Tail) for estimating the GP distribution. Figure 2.2c shows that TellTail provides accurate confidence estimates: on a Twitter dataset of football clubs, TellTail separates ground truth football clubs (red triangles) from random subgraphs (blue dots), while providing confidence estimates. The 'empirical quantiles' plot the 90% and 99% quantiles of the measure on random subgraphs, matching the theoretical quantiles and verifying that our confidence estimates are accurate.



(a) Accuracy (UCForum dataset)



(b) Linear runtime and speedup given



(c) Accurate confidence estimates

Figure 2.2: (a) **Accuracy:** TellTail outperforms comparable baselines in accuracy in detecting injected blocks. (b) **Scalability:** slopes parallel to the dotted lines indicate linear growth. TellTail was $140,000$ times faster than the more naive sampling-based approach. (c) **Accurate confidence estimates:** our measure clearly separates ground truth subgraphs (red triangles) from random subgraphs (blue dots) while providing confidence estimates.

2.2 Fraudar: Fraud Detection in an Adversarial Setting

Section based on work that appeared at KDD16 [HSB+16b].

Motivation Given a bipartite graph of users and the products that they review, or followers and followees, how can we detect fake reviews or follows? Existing fraud detection methods (spectral, etc.) try to identify dense subgraphs of nodes that are sparsely connected to the remaining graph. Fraudsters can evade these methods using *camouflage*, by adding reviews or follows with honest targets so that they look “normal”. Even worse, some fraudsters use *hijacked accounts* from honest users, and then the camouflage is indeed organic. Our focus is to spot fraudsters in the presence of camouflage or hijacked accounts, using an algorithm that is robust against adversaries, provides upper bounds on the effectiveness of fraudsters, and is effective in real-world data.

Problem 2.2: Robust Fraud Detection

- **Given:** bipartite graph $G = (V, E)$, e.g. nodes corresponding to users and products, and edges corresponding to reviews
- **Output:** detect the subgraph containing fraudulent attacks, in a way that minimizes the number of edges that fraudsters can add without being detected.

2.2.1 Main Ideas

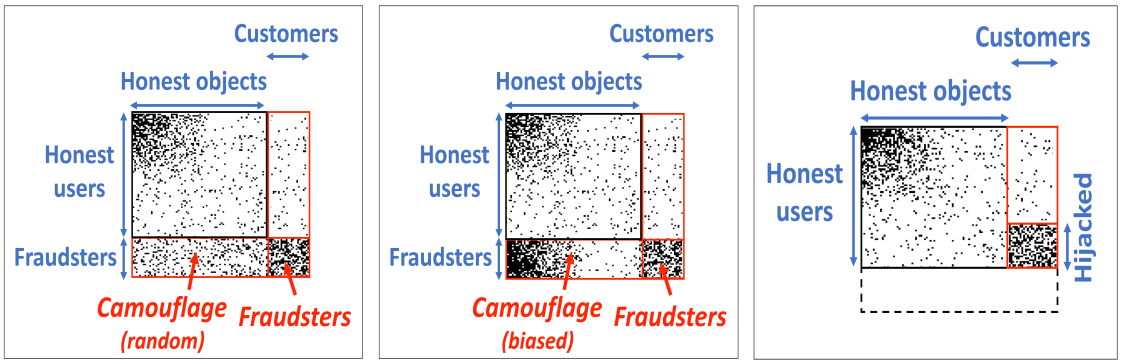


Figure 2.3: **Three examples of possible attacks**: (a) random camouflage; (b) biased camouflage; (c) hijacked accounts.

Attack Model We assume that fraudsters are hired to use users they control to add edges pointing to a subset of nodes in $\mathcal{W}$. For example, a business may pay for followers on Twitter or positive reviews on Yelp. In general, fraudsters add a large number of edges, inducing a dense subgraph between the fraudster accounts and customers, as shown in the bottom right corner of each subplot of Figure 2.3.

To mask the fraud, fraudster accounts can add arbitrary "camouflage", i.e. edges pointing from their user accounts to any of the nodes in $\mathcal{W}$ that are not customers. We assume that fraudsters have *complete* knowledge of the graph and fraud detection mechanisms, enabling worst-case camouflage for any fraud detection system we create. Examples of the possible types of camouflage are given in Figure 2.3: (a) adding camouflage edges to random honest users, (b) camouflage biased toward high degree nodes, (c) using hijacked accounts, whereby fraudster accounts have realistic patterns of camouflage essentially similar to that of honest users.

While it is trivial for fraud accounts to add edges to any other node, it is more difficult for customer accounts to get honest edges. In particular, we assume that a customer would try to increase their number of incoming edges by a significant fraction, and as a result a large fraction of their incoming edges will be from fraudsters.

Method Intuitively, note that adversaries can only add camouflage to the rows with fraudulent accounts, not the columns. This suggests the use of column-weighting in order to detect adversaries: we weight the graph edges such that columns with fewer nonzero entries receive higher weights. Since fraudsters have a large fraction of their edges from fraudsters, their edge count should be high relative to their column sum, and so this column weighting should assign them high weight.

Moreover, this column weighting approach is robust to manipulation, as no matter how much row camouflage a fraudster adds, they cannot reduce their weight, which is a function of the columns. This then limits the number of fraudulent edges they can add before they are detected.

With this intuition, we propose using metrics g of the following form, which add up the amount of 'node suspiciousness' $f_{\mathcal{V}}$ and 'edge suspiciousness' $f_{\mathcal{E}}$ in a subgraph $\mathcal{S}$, and divide it by the size of the subgraph:

Definition 2.1: FRAUDAR Metric

The suspiciousness of subgraph $\mathcal{S}$ is:

$$g(\mathcal{S}) = \frac{f(\mathcal{S})}{|\mathcal{S}|} \tag{2.1}$$

where total suspiciousness f is:

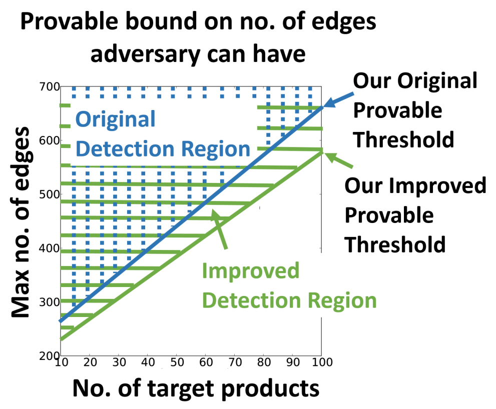
$$\begin{aligned} f(\mathcal{S}) &= f_{\mathcal{V}}(\mathcal{S}) + f_{\mathcal{E}}(\mathcal{S}) \\ &= \sum_{i \in \mathcal{S}} a_i + \sum_{i,j \in \mathcal{S} \wedge (i,j) \in \mathcal{E}} c_{ij}, \end{aligned} \tag{2.2}$$

The node suspiciousness is a sum of terms a_i corresponding to the nodes in $\mathcal{S}$, which can be thought of as how individually suspicious that particular user or object is, while the edge suspiciousness is a sum of constants c_{ij} corresponding to the edges in between $\mathcal{S}$, which can be thought of as how suspicious that particular edge is. Following our discussion on column weighting, we set c_{ij} to various decreasing functions of the respective column sum, in order to downweight normal nodes with naturally high degree, and instead catch fraudulent subgraphs. See the paper for the full details on these scores.

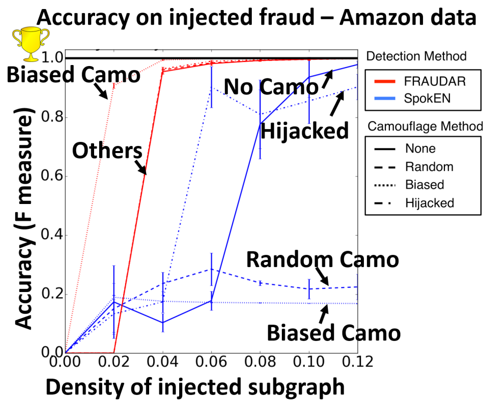
This form has a number of advantages: we show that it can be optimized greedily in a way that is (a) scalable; (b) offers theoretical guarantees, and (c) is robust to camouflage.

2.2.2 Results

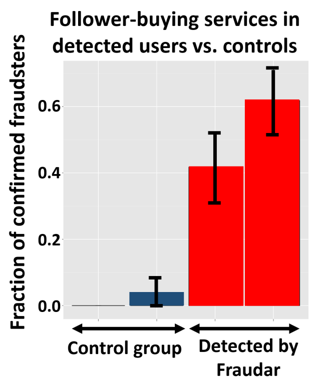
Figure 2.4a shows our thresholds and the improvement due to our modified approach on an Amazon purchasing network. Our optimizations improve the (data-dependent) bounds by lowering it to the green region, meaning that the adversary can only add fewer edges before they are caught. Figure 2.4b compares our approach in terms of F-measure to a baseline fraud detection approach, on simulated attacks on the Amazon data, in which we inject a dense subgraph



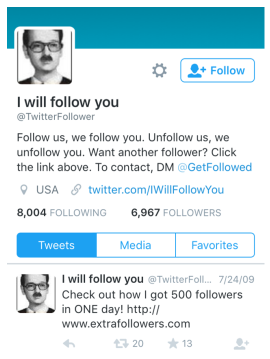
(a)



(b)



(c)



I will follow you
@TwitterFollower

Follow us, we follow you. Unfollow us, we unfollow you. Want another follower? Click the link above. To contact, DM @GetFollowed

USA twitter.com/IWillFollowYou

8,004 FOLLOWING 6,967 FOLLOWERS

I will follow you @TwitterFoll... 7/24/09
Check out how I got 500 followers in ONE day! http://www.extrafollowers.com

(d)

Figure 2.4: **(a) Our theoretical thresholds:** fraudsters in the detection region will be caught by our approach. **(b) Accuracy:** Fraudar outperforms baselines. **(c) Confirmed fraudsters**: a large fraction of our detected followees (left red bar) and followers (right red bar) compared to almost none among non-flagged accounts (2 control groups). Confirmation was done by inspecting Tweets that advertise '*TweepMe*' or *TweeterGetter*. **(d) Real-life results:** a sample fraudster caught.

consisting of fraudulent nodes. Our approach outperforms the baseline on a variety of camouflage settings, and is more robust against camouflage.

In addition, we ran our algorithm on a 1.47 billion edge Twitter follower-followee graph, which detected a block of around 4000 detected followers and around 4000 followees. Figure 2.4c shows that a majority of the detected followees are confirmed fraudsters, where confirmation was done by inspecting tweets that advertise '*TweepMe*' or '*TweeterGetter*' (not including other similar services). Figure 2.4d shows a sample fraudster caught.

Our theoretical guarantee is as follows, and implies our algorithm will detect a fraudulent block without fail if it contains more edges than this data-dependent threshold:

Theorem 2.1

Let $(\hat{\mathcal{A}}, \hat{\mathcal{B}})$ be the block detected by FRAUDAR. Then the number of edges that a fraudulent block of size (m_0, n_0) can have without being detected is at most $2(m_0 + n_0)g(\hat{\mathcal{A}} \cup \hat{\mathcal{B}}) \log(m_0/\lambda + c)$.

Chapter 3

Time Series

How can we use temporal information to detect time periods containing unusual activity?

Temporal data is commonplace. Data arising from online systems often takes the form of individual events associated with timestamps: e.g., timestamped tweets, follows, purchases, likes, etc. In contrast, in the traditional time-series setting, we have the values of some real-valued sensor that varies over time: e.g. amount of a pollutant over time, voltage sensors in a power grid, etc. In both these cases, temporal information is of key importance, as unusual activity often takes the form of either a major change in behavior pattern, or a short period of highly unusual activity.

In this chapter, we consider how to detect time periods with unusual activity. BirdNest focuses on a prototypical case involving categorical data, arising from online systems: here we have ratings data, where users submit ratings for products over time, and we want to detect suspicious users performing rating manipulation. StreamCast focuses on the more traditional real-valued sensor data, in a power grid setting: here our goal is to find anomalies in the context of a physics-based model for power systems data. Finally, BNB provides a nonparametric anomaly detection approach that can be used on categorical, numeric or ordinal data, without making assumptions about the distribution that the data comes from, as well as a streaming approach for such data.

3.1 BirdNest: Fraud Detection in Timestamped Ratings

Section based on work that appeared at SDM16 [HSB+16a].

Motivation How do we detect fraudulent users in a dataset containing timestamped ratings by a set of users for various products? In this section we propose our Bayesian model, Bird (Bayesian Inference for Ratings Data), and our suspiciousness metric, Nest (Normalized Expected Surprise Total), a principled likelihood-based suspiciousness metric for fraud detection. Our method provides a principled way to combine rating and temporal information to detect rating fraud, and to find a tradeoff between users with extreme rating distributions vs. users with larger number of ratings.

Problem 3.1

- **Given:** a set of users and products, and timestamped ratings (e.g. 1 to 5 stars) by users for products,
- **Output:** a suspiciousness score for each user.

3.1.1 Main Ideas

We use the intuition that 1) fraudulent ratings are **polarized**: normal ratings have a relatively smooth distribution, while fraudulent ratings are very positive (5 stars) or very negative (1 star), since their goal is either to promote or sabotage a product. Moreover, fraudulent ratings tend to be either bursty or excessively regular (e.g. exactly every 24 hours) which are both characteristics of bots.

To capture these, we first preprocess each rating by computing its **time difference** from the previous rating, i.e. the difference between its timestamp and the timestamp of the last rating given by the same user. We then bucket the time differences logarithmically. We can then model this bucketed data using a discrete, i.e. multinomial approach, allowing us to flexibly detect deviations from normal behavior without assuming a more restrictive parametric form, such as a Gaussian distribution. We treat each user as a histogram both over the possible rating values (1 to 5 stars) and these bucketed time differences. For example, if Alice gave 100 5-star ratings, all of which were 1 second apart, her rating histogram and time difference histogram would be much more skewed than the rest of the users, which should intuitively give her a low likelihood under a Bayesian model.

Hence, to model the data, we use a Bayesian mixture model in which each user belongs to one of K clusters: in general, there is no single type of user behavior, so we use clusters to capture different types of user behavior. Each cluster represents a certain type of rating distribution and temporal distribution for the users in that cluster. However, even within a single cluster, it would not be reasonable to expect all users to behave exactly the same way. Thus, instead of using a single rating/temporal distribution per cluster, we allow small deviations per user. To do this, we represent each cluster by a Dirichlet prior. Each user belongs to a cluster, in which case the user is represented by a multinomial distribution drawn from this Dirichlet prior. Finally, each of this user's ratings is drawn from this multinomial distribution.

3.1.2 Results

We ran our algorithm on data from the Flipkart e-commerce platform, and verified the results against ground truth fraudulent users, finding that all of the top 50 users flagged by BirdNest were indeed involved fraud, and 211 of the top 250 users were involved in fraud.

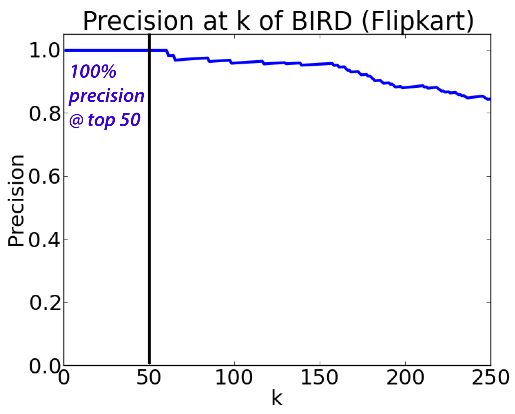
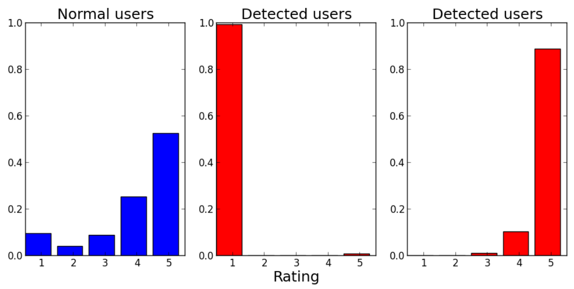
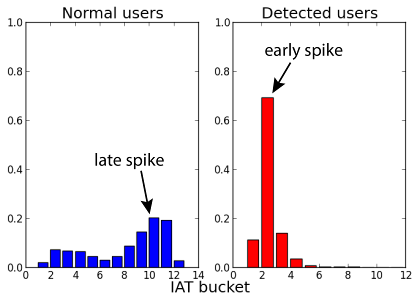


Figure 3.1: **BIRDNEST is effective in practice**, with 211 users of the top 250 flagged by BIRDNEST involved in fraud.

A common pattern that the domain-experts at Flipkart found was that most of the users labeled as fraudulent are either spamming 4/5 star ratings to multiple products from a single seller (boosting seller's ratings), or spamming 1/2 star ratings to multiple products from another seller (defaming the competition). Figure 3.2b shows that while normal users have a typically smooth J-shaped distribution, the detected users consist of two groups: highly negative users (middle) and highly positive users (right).



(a) Ratings of normal users (left) vs detected users, which consist of highly negative users (middle) and highly positive users (right)



(b) Inter-arrival times of normal users (left) vs detected users (right).

Figure 3.2: Common patterns observed that detected users' ratings are (a) highly polarized, and are (b) more bursty than normal users.

3.2 StreamCast: Modeling Power Grid Time Series

Section based on work that appeared at SDM18 [HSP$^+$18].

Motivation How do we forecast the power consumption of a location for the next few days, and detect anomalies in the same data? Our goal is to model power consumption data in a way that captures seasonal patterns, while being based on a physics-based model of electrical behavior, which improves the accuracy of forecasting. The online forecasting problem is closely tied to anomaly detection, since time points where large differences exist between the forecasted and actual values can be flagged as anomalies.

Hence, our goal is to forecast voltage V and current I for the future. V consists of real and imaginary parts, denoted V_r and V_i (and similarly for current). Then our forecasting problem is:

Problem 3.2: Multi-step Forecasting

- **Given:** values of current $(I_r(t), I_i(t))$, voltage $(V_r(t), V_i(t))$, and temperature $T(t)$ for $t = 1, \cdots, N$, and given temperature forecasts for the next N_f time steps (i.e. for $t = N+1, \cdots, N+N_f$),
- **Forecast:** voltage and current for N_f time steps in the future; i.e. $(V_r(t), V_i(t))$ and $(I_r(t), I_i(t))$ for $t = N+1, \cdots, N+N_f$.

3.2.1 Main Ideas

Our main idea is to start with the physics-based (but non-temporal) BIG model [JPS+17], and modify it into a temporal model. We find that using physics-based models provides higher forecasting accuracy than generic time series approaches, and also allows for interpretability of individual parameters. The BIG model treats current as a linear function of the voltage, parameterized by the BIG parameters: susceptance (B), conductance (G), and a current offset (α_r, α_i).

$$\begin{aligned} I_r(t) &= G \cdot V_r(t) - B \cdot V_i(t) + \alpha_r + \text{noise} \\ I_i(t) &= B \cdot V_r(t) + G \cdot V_i(t) + \alpha_i + \text{noise} \end{aligned} \tag{3.1}$$

Here G can be interpreted as the component contributing to real power consuming loads (e.g. due to light-bulbs), while B can be interpreted as the contribution of the reactive power component (e.g. due to motors or capacitors).

Temporal BIG Model Eq. (3.1) is a linear equation, where I_r, I_i, V_r and V_i are observed, and G, B, α_r, α_i are unknown parameters, which could be estimated using linear regression. However, rather than treating them as constants, we want to model them as varying over time, treating them as time series: i.e. $G(t), B(t), \alpha_r(t), \alpha_i(t)$. For example, G and B, which represent types of electrical load, should actually vary seasonally over the day as usage of lightbulbs, etc. varies.

We thus define a seasonal model for G and B (see the paper for full details), which we estimate in a streaming manner using a gradient descent approach. (Note that standard gradient descent would not capture seasonal patterns and is not an online algorithm, so we modify the gradient update steps accordingly.) We then consider how to use temperature as an input to improve forecasting, with the intuition that higher temperatures generally lead to increased power load due to the use of air-conditioning (we observe this empirically in real data).

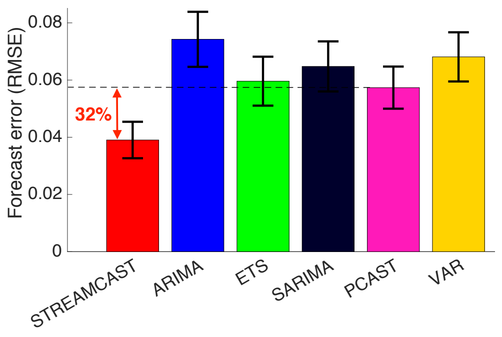
Anomaly Detection Finally, given our forecast, we compute the forecasting error for current, $E_r(t) = I_r(t) - \hat{I}_r(t)$, and $E_i(t) = I_i(t) - \hat{I}_i(t)$. Using this, we define the anomalousness at time t as the total number of interquartile ranges that the data deviates from the forecast, at time t:

Definition 3.1: Anomalousness

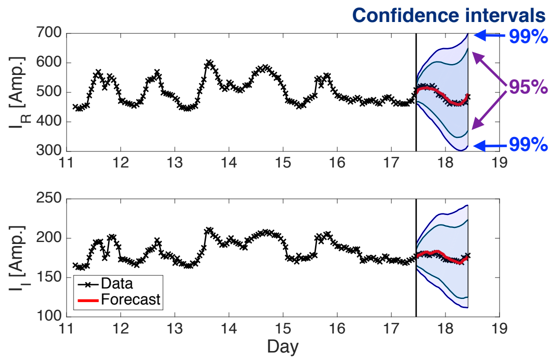
The anomalousness at time t is:

$$\text{anomalousness}(t) = \frac{E_r(t)}{\text{IQR}(E_r(1:N))} + \frac{E_i(t)}{\text{IQR}(E_i(1:N))}$$

3.2.2 Results



(a) Forecasting accuracy



(b) Confidence intervals

Figure 3.3: Forecasting results of StreamCast.

We evaluate the multi-step forecasting accuracy in Root Mean Square Error (RMSE) of StreamCast on real power grid voltage and current data. Figure 3.3a shows that StreamCast outperforms baselines in accuracy, and Figure 3.3b shows that it provides confidence intervals.

Figure 3.4 shows anomaly detection results, where our method outputs a plot ('anomalousness') of the anomaly score over time. There is a large spike in anomalousness around day 2; note that the anomaly is not at all visible from only the current time series. In the voltage time

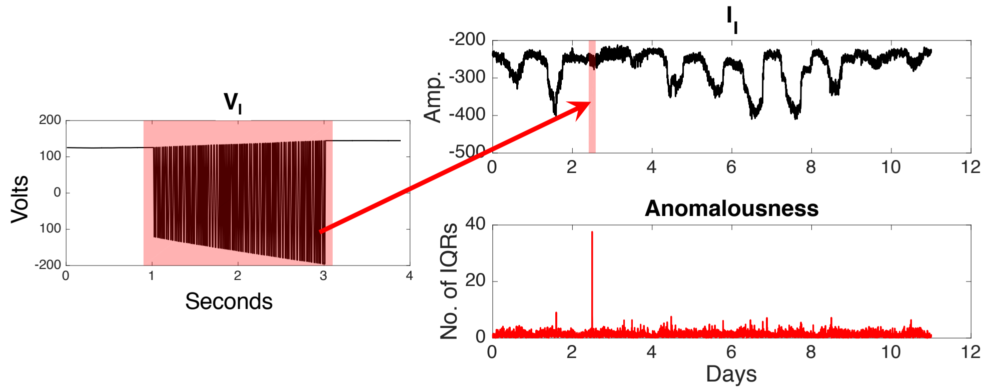


Figure 3.4: **StreamCast detects an anomaly:** it corresponds to a 2-second period where voltage rapidly oscillates between negative and positive values.

series, however, we find a 2-second period of quick oscillations between negative and positive values exactly at the time of the anomaly. Follow-up analysis reveals that the oscillation is likely an error in the measuring device: the complex angle of voltage is synchronized with the rest of the system via a GPS signal, and in case of loss of this GPS signal, we may observe oscillations such as those in Figure 3.4, which explains why the voltage magnitude remains stable while the angle oscillates.

3.3 BNB: Change Detection in Multivariate Time Series

Section based on work that is under review.

Motivation Given multivariate time series data, how do we detect changes in the behavior of the time series: for example, the onset of illnesses or complications in patients? More challengingly, how do we detect changes without making strong assumptions about the type of model that the data follows before and after the change? We propose BNB (Branch and Border), an online, nonparametric change detection method that detects multiple changes in multivariate data, allowing for categorical, numerical, and ordinal data.

Thus, our goal is an online, nonparametric change detection algorithm. Informally, the problem we aim to solve is:

Problem 3.3: Online Change Detection

- **Given** a multivariate stream of d-dimensional data $X_1, X_2, \cdots$ (possibly numerical, ordinal or categorical data);
- **Find** a set of time ticks t where changes occurred, reported in a streaming manner.

3.3.1 Main Ideas

The main idea of our approach is its use of random partitions to measure how good a change is. We start by introducing the intuition for this approach.

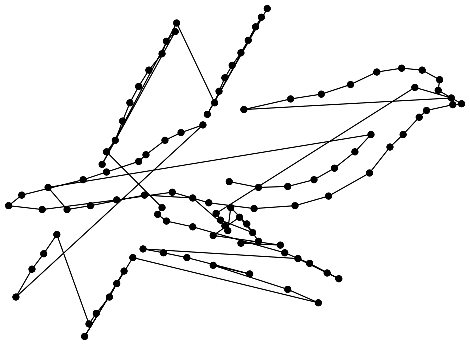
First consider a 1-dimensional toy dataset: $X = [0, 1, 0, 0, 10, 10, 9, 10]$, which clearly has a change point at $t = 5$. We use the idea that if a good change point is present, a **random partition** has a high change of cleanly separating the points before and after the change point. For the toy dataset, if we sample a random cut point uniformly between 0 and 10 (the min and max of the data), then with 80% probability, the cut point lies in $(1, 9)$. In this case, the cut point perfectly separates the points before and after $t = 5$ (since the first 4 data points are less than the cut point, while the last 4 are greater). Intuitively then, $t = 5$ is a good change point because the points before and after (or equal to) it are easily separable by random partitions.

In real data, single cuts do not always suffice, and we may need multiple cuts, but the intuition is similar. Consider the drawing of a bird on a branch in Figure 3.5a, containing an unknown change point, which is when we moved from drawing the branch to drawing the bird. The connecting line segments (in black) indicate that the drawing is a time series, not just a collection of points.

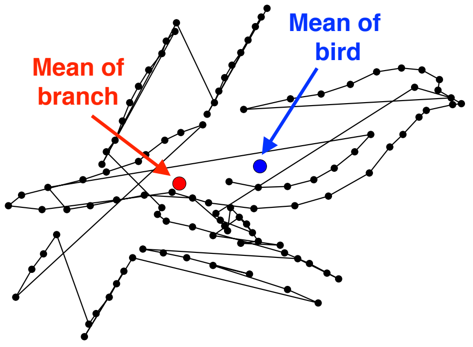
Since the points in the bird and the branch intuitively have different distributions, the bird and branch should be separable, but this is challenging: e.g. the bird's feet and tail are particularly hard to separate from the branch. Indeed, Figure 3.5b shows that the standard approach of fitting distributions (Gaussian or otherwise) before and after the change has difficulty separating the bird from the branch: these approaches rely on a large difference in means before and after the change, but the bird and branch have fairly close means, and the difference between the two is swamped by the large amount of variance within each class. Indeed, more generally, these approaches make two strong assumptions: 1) the data is well-modeled by that type of distribution, and 2) the data on each side of the change are independent, i.e. no autocorrelation over time is present. In realistic scenarios, these do not hold, and can have adverse effects: e.g. if outliers are present in the data, they can overly influence the fitted distributions, which can cause inaccuracy.

Figure 3.5c illustrates our random partition approach. We conduct a series of random cuts (gray lines) that partition the space recursively. At the time of the correct change, the points before the change (in red) and after the change (in blue) can be cleanly separated by the random partition: i.e. all boxes contain either only red or only blue points. The stronger the change point, the easier the red and blue points are to cleanly separate, and the simpler the random partition we would need to cleanly separate them. Hence, in Figure 3.5d we compute a score for each possible change point based on how easily we can separate the points before and after the change using random partitions. Averaging over many such partitions, Figure 3.5d shows that the highest change score indeed coincides with the true change point.

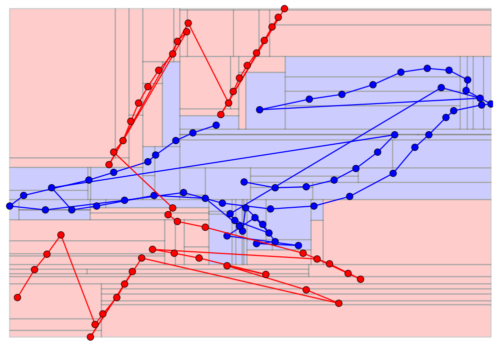
This explains our method's name (Branch and Border): a good change point is one that allows for a relatively clean separation between points before and after the change. Partitioning the space recursively can be thought of as growing a tree; if this is a good change point, the points before and after the change should be separated into different 'branches' of this tree. The random cuts used to grow the tree (i.e. the gray lines in Figure 3.5c) are then 'borders' that separate the points before and after the change.



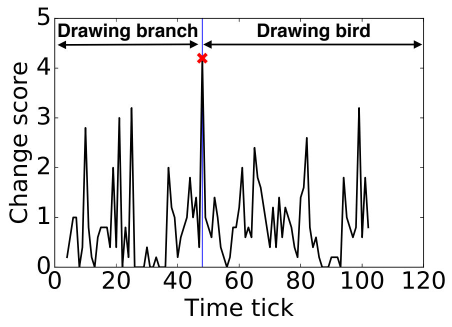
(a) **Drawing of bird and branch**



(b) **Gaussian approach fails**



(c) **Random partition approach finds change point**



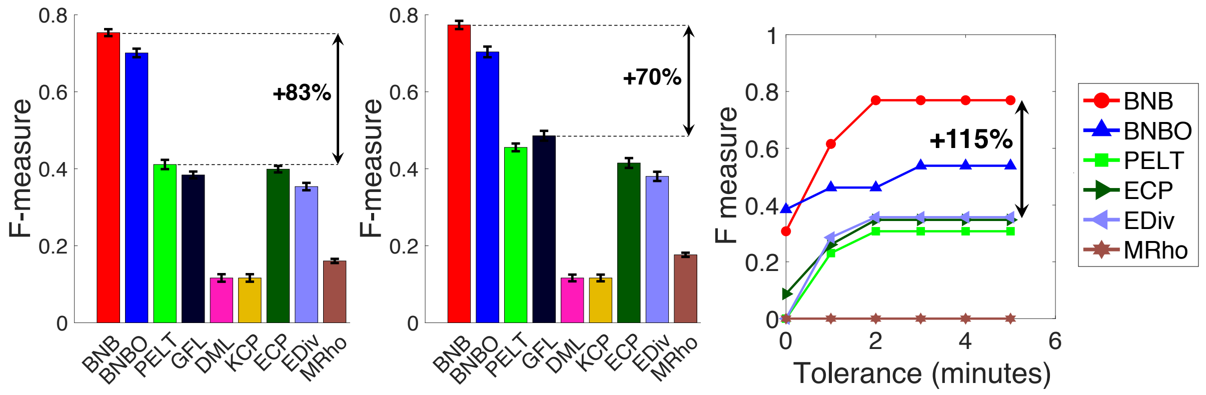
(d) **Peak in change score coincides with ground truth**

Figure 3.5: **BNB makes use of arbitrarily-shaped regions:** (a) A drawing of a bird on a branch, with an unknown change point (when moving from the branch to the bird). (b) The bird and the branch have very similar means, making it difficult to separate them by fitting Gaussian (or most other) distributions. (c) At the time of the true change, the red points (before the change) and blue points (after the change) can be fully separated using a relatively simple random partition, suggesting that this is a good change. Note that the blue region (i.e. boxes only containing points after the change) conforms to the bird's outline well despite the bird's unique shape which is hard to separate from the branch. (d) Averaging over many such partitions, indeed the change score is highest at the true change time.

3.3.2 Results

We compare the accuracy of BNB, and its online variant, BNBO, in detecting injected multiple change points on 11 datasets, in terms of F-measure. The results averaged over all 11 datasets are shown in Figures 3.6a for unnormalized data and 3.6b for normalized data. We see that our methods both outperform the baselines significantly, by 70% or more F-measure in the case of BNB.

We also evaluate our methods on a real-world occupancy detection task, with ground truth change points representing points where the room changed from being unoccupied to occupied, or vice versa, resulting in 13 change points. The sensors measure the room's temperature, humidity, light and carbon dioxide levels. Ground truth occupancy was obtained from time-



(a) Without normalization (b) With normalization (c) Occupancy detection

Figure 3.6: **Accuracy for multiple change detection:** (a) F-measure of detecting multiple change points compared to baselines, averaged over 11 datasets. Error bars indicate one standard deviation. (b) Same accuracy results, except with normalization for all methods. (c) Accuracy for a real-world occupancy detection task, where changes refer to the room changing from occupied to unoccupied, or vice versa.

stamped pictures taken every minute. We evaluate each algorithm based on F-measure, where reported change points are considered as correctly matched to a given ground truth change point if the two are at most ‘*tolerance*’ minutes apart, where we show results for different values of ‘*tolerance*’ plotted in the x-axis of Figure 3.6c.

Chapter 4

Dynamic Graphs

Given time-varying sensor data arranged on a graph, how can we detect the time and location of unusual events?

In this chapter, we consider graph-structured time series data. In particular, we have a set of sensors arranged in a graph, each of which collects data over time. This type of data is common in real-world sensor datasets: traffic speed sensors are arranged on a road network, and voltage sensors are arranged on a power grid network. Spatiotemporal datasets, e.g. air quality sensors placed at geographical locations, can also be coverted to graph-structured time series, by constructing the graph joining each sensor to its nearest neighbors.

We develop algorithms for detecting critical or anomalous events, such as traffic accidents or power line failures. Our goal is to generally to detect both the time and location of the event. Our ChangeDAR algorithm treats this as a change detection problem: it assumes that a change occured at a certain time which affects a connected subgraph of sensors, and our goal is to detect this time and subgraph in an online manner. Our GridWatch [HES+18] approach adds to this by also studying how to near-optimally select locations for sensors to be placed on a power grid graph, improving the detection of electrical component failures on a power grid graph by 59% or more F-measure.

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

Section based on work that is under review.

Motivation 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.

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

Problem 4.1: 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.

4.1.1 Main Ideas

Our main idea is to apply the Prize-Collecting Steiner Tree (PCST) framework to our setting. The PCST framework [BGSLW93] 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 PCST aims to construct a road network to maximize net profit (or minimize net cost).

In our setting, at each time tick, we define 'prizes' on nodes representing some description-length based score for how strongly we believe a change point occurred at that node. The score assigned to a node is the number of bits we save by assuming that a change occurred on that node. Then, the PCST framework allows us to search for a subgraph of nodes which are connected to one another, and on which the data provides evidence that a change point jointly occurred on these nodes at a certain time.

4.1.2 Results

Using the PCST framework, we get constant-factor approximation guarantees:

Theorem 4.1: Approximation Guarantee

Our algorithm provides an approximation guarantee of 2 for optimizing our (description-length based) objective f_t: letting $\mathcal{S}'$ be the returned set and $\mathcal{S}^*$ the optimal solution:

$$f_t(\mathcal{S}') \leq 2 \cdot f_t(\mathcal{S}^*). \tag{4.1}$$

Traffic Data Figure 4.1 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. 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.

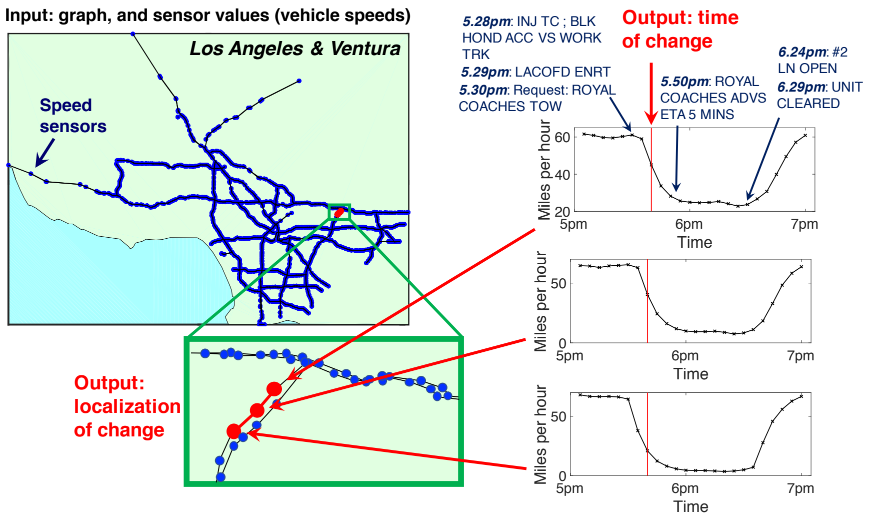

Input: graph, and sensor values (vehicle speeds)
Los Angeles & Ventura
Speed sensors
Output: localization of change
5.28pm: INJ TC ; BLK HOND ACC VS WORK TRK
5.29pm: LACOFD ENRT
5.30pm: Request: ROYAL COACHES TOW
Output: time of change
5.50pm: ROYAL COACHES ADVS ETA 5 MINS
6.24pm: #2 LN OPEN
6.29pm: UNIT CLEARED
Miles per hour
Time
5pm
6pm
7pm

Figure 4.1: **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).

Figure 4.2 shows that ChangeDAR outperforms several baselines in precision and recall of locating traffic accidents, by 70% and 227% respectively, and F-measure by 124%.

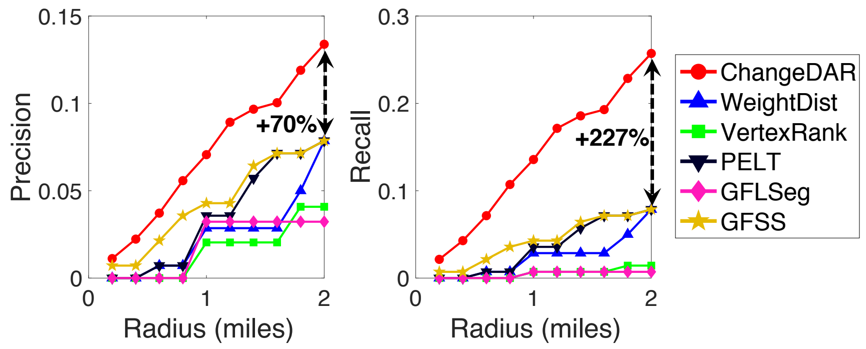


Figure 4.2: **ChangeDAR accurately locates traffic accidents**: ChangeDAR outperforms baselines in precision (left) and recall (right) on ground truth traffic accidents. The x-axis plots the radius used to determine whether a change point output by each algorithm matches a ground truth accident.

4.2 GridWatch: Sensor Placement and Anomaly Detection in the Electric Grid

Section based on work to appear in ECML-PKDD18.

Motivation Given sensor readings over time from a power grid consisting of nodes (e.g. generators) and edges (e.g. power lines), how can we most accurately detect when an electrical component has failed? More challengingly, given a limited budget of sensors to place, how can we determine where to place them to have the highest chance of detecting such a failure? Maintaining the reliability of the electrical grid is a major challenge. An important part of achieving this is to place sensors in the grid, and use them to detect anomalies, in order to quickly respond to a problem. Our goals are to design 1) an online anomaly detection algorithm given a fixed set of sensors; and 2) to find near-optimal positions for placing new sensors on a graph, that maximizes the probability of detecting an anomaly.

Hence, our goal is an online anomaly detection algorithm:

Problem 4.2: Online Anomaly Detection

- **Given:** A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a subset $\mathcal{S}$ of nodes which contain sensors. For each sensor, we have a continuous stream of values of real and imaginary voltage $V(t)$ and current $I(t)$ measured by these sensors.
- **Find:** At each time t, compute an anomalousness score $A(t)$, indicating our confidence level that an anomaly occurred (i.e. a transmission line failed), and the location of the most likely anomaly.

For cost reasons, it is generally infeasible to place sensors at every node. Hence, an important follow-up question is where to place sensors so as to maximize the probability of detecting an anomaly.

Problem 4.3: Sensor Placement

- **Given:** A budget k of the number of sensors we can afford, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a simulator that allows us to simulate sensor readings at each node.
- **Find:** A set of nodes $\mathcal{S} \subseteq \mathcal{V}$, which are the locations we should place our sensors, such that $|\mathcal{S}| = k$.

4.2.1 Main Ideas

Online Anomaly Detection Our anomaly detection algorithm uses a domain-dependent approach which exploits the fact that electrical sensors consist of a voltage reading at a node as well as the current along each adjacent edge.

As an illustrative example, consider the simple power grid shown by the graphs in Figure 4.3. The power grid consists of a single generator, a single load, and power lines of uniform resistance. When the edge marked in the black cross fails, current is diverted from some edges to others, causing some edges to have increased current flow (blue edges), and thus increased power, and others to have decreased current flow (red edges). Current flows are computed using a standard power grid simulator, Matpower [ZMST11].

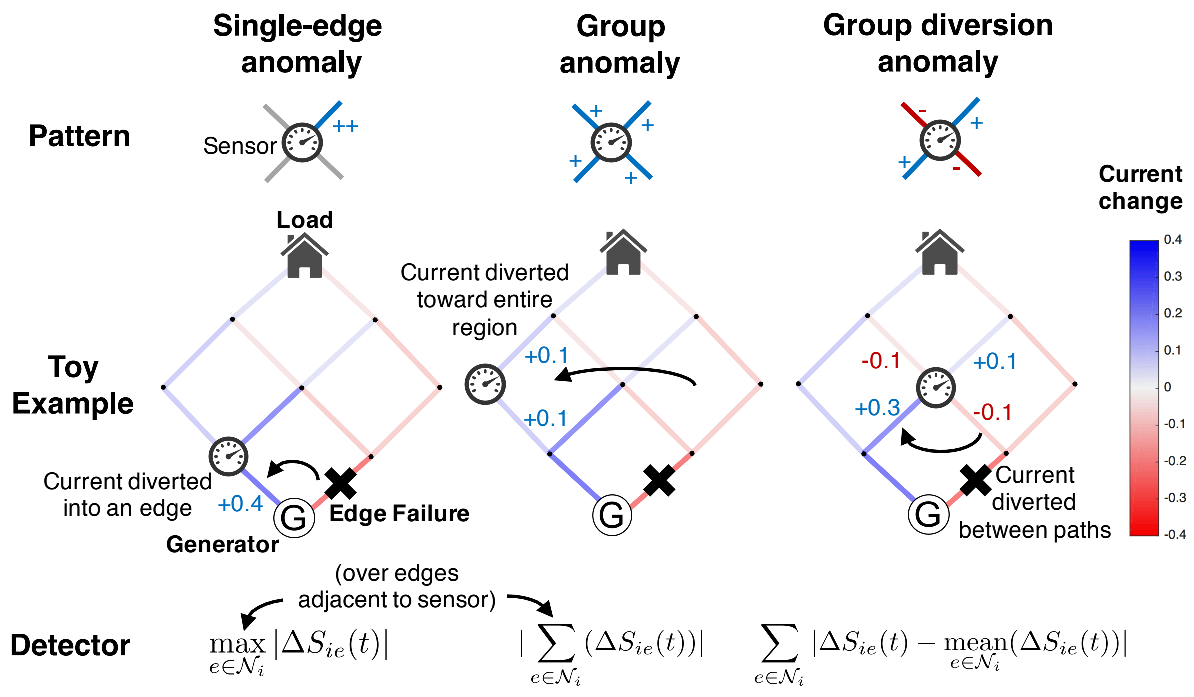

Single-edge anomaly
Group anomaly
Group diversion anomaly
Pattern
Sensor
++
+
-
Current change
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
Load
Toy Example
Current diverted toward entire region
+0.1
-0.1
+0.3
Current diverted into an edge
+0.4
G
Edge Failure
Generator
Current diverted between paths
(over edges adjacent to sensor)
Detector
$\max_{e \in \mathcal{N}_i} |\Delta S_{ie}(t)|$
$|\sum_{e \in \mathcal{N}_i} (\Delta S_{ie}(t))|$
$\sum_{e \in \mathcal{N}_i} |\Delta S_{ie}(t) - \operatorname{mean}_{e \in \mathcal{N}_i}(\Delta S_{ie}(t))|$

Figure 4.3: **Domain-aware model for anomalies:** edge failures divert current from one region of the graph to another, forming 3 patterns. Edge color indicates change in current due to the edge failure: blue is an increase; red is a decrease. *Left:* current from the deleted edge diverts into an edge, resulting in a highly anomalous value along a single edge. *Center:* the deletion diverts current between areas (from the right side to the left side of the graph), forming a group anomaly at the leftmost sensor, due to its multiple positive edges. *Right:* the deletion diverts current between paths, making the central sensor have multiple positive and negative edges.

In the left plot, the edge deletion results in a large amount of current flowing from the generator through its left edge instead of its right edge. This results in a **single edge** with a large anomalous value, which can be detected by the sensor by taking the maximum of the deviation along its neighboring edges.

In the middle plot, the edge deletion cuts off a large amount of current that would have gone from the generator toward the right side of the graph, diverting it into the left side of the graph.

This results in some nodes in the left region with all their neighboring edges having positive changes (blue), resulting in a **group anomaly** such as the leftmost node.

In the right plot, the edge deletion diverts current between nearby edges. In particular, current diversions around the central node cause it to have neighbors which greatly differ from each other: 2 positive edges and 2 negative edges, resulting in a **group diversion anomaly** with edge values around the sensor deviating greatly from one another.

Hence, we use 3 detectors, each of which is designed to detect one of these 3 types of anomalies. We evaluate these detectors as the data arrives in a streaming manner, and report an anomaly if detector on a node exceeds a fixed threshold value, measured in terms of inter-quartile ranges.

Sensor Selection To select locations for sensors to place given a fixed budget of k sensors to place, our main idea is to construct an optimization objective for the anomaly detection performance of a subset $\mathcal{S}$ of sensor locations, and show that this objective has the 'submodularity' property, showing that a greedy approach gives approximation guarantees.

4.2.2 Results

Approximation Guarantee The nondecreasing and submodularity properties of our objective imply that our algorithm achieves at least $1 - 1/e$ ($\approx 63\%$) of the value of the optimal sensor placement. Letting f be our objective, $\hat{\mathcal{S}}$ be the set returned by our algorithm and $\mathcal{S}^*$ be the optimal set:

Theorem 4.2

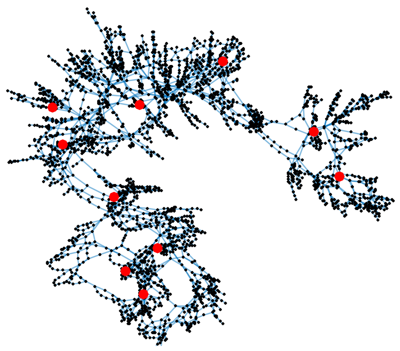
Our algorithm has a constant-factor approximation guarantee: i.e.

$$f(\hat{\mathcal{S}}) \geq (1 - 1/e) f(S^*) \tag{4.2}$$

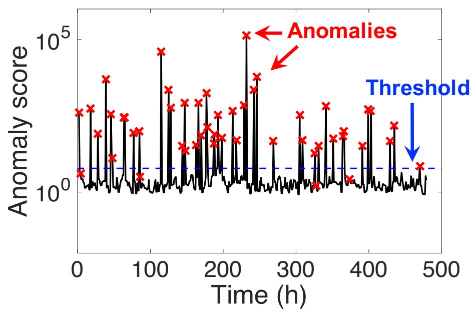
Experiments To evaluate the accuracy of our methods, we use a power grid simulator, Matpower [ZMST11] to generate power grid data on graphs from real power grids. We inject anomalies, involving the breakdown of a power line (i.e. edge), and test the accuracy of our algorithm and other approaches for detecting these anomalies.

Figure 4.4a shows an example of the sensors selected by GridWatch: red circles indicate positions chosen. Note that the sensors are spread over different areas of the graph and each sensor is centrally located within an area, to maximize the probability of detecting an anomaly.

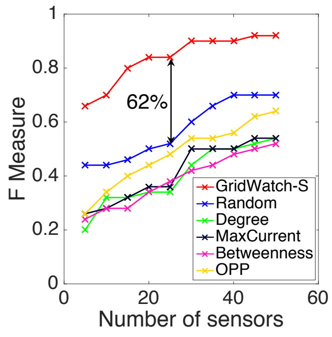
Figure 4.4b shows the anomaly scores (black line) output by GridWatch, which accurately match the ground truth. Figure 4.4c shows that the sensor selection approach in GridWatch outperforms baseline selection approaches on a real power grid graph, in terms of F-measure of correctly detecting the time at which an anomaly occurred.



(a) **Selects sensor locations**



(b) **Anomaly scores**



(c) **Accuracy**

Figure 4.4: (a) GridWatch selects sensor locations on a real power grid graph, with a constant factor approximation guarantee. Red circles indicate positions chosen for sensors. (b) GridWatch computes anomaly scores (black line). Red crosses indicate ground truth - notice 100% true alarms (all black spikes above blue line are true alarms) and only 4 false dismissals (red crosses below blue line). (c) F-measure of GridWatch compared to baselines.

Chapter 5

Related Work

5.1 Graphs

Dense Subgraph Measures Average degree [Gol84] is a common measure that can be optimized exactly [Gol84] or via $1/2$-approximation factor greedy algorithms [Cha00]. Variants allow for size restrictions [AC09] or local subgraphs [And10]. Other measures include edge density, which underlies quasi-cliques [ARS02], edge surplus [TBG+13], triangle and k-clique density [Tso15], discounted average degree [YH18], and minimum internal degree, which defines k-cores [SERF16]. Related measures underlie k-plexes [SF78] and k-trusses [Coh08]. [ANMJZ12] evaluates density by subtracting the expected density of similar clusters.

Among probabilistic approaches, many generative models for graphs exist, such as scale-free graphs [LADW05], Kronecker graphs [LCK+10], and so on. However, computing surprisingness under these models is often infeasible because the models do not allow for tractable analysis. For the Erdos-Renyi model, [BE76] analyzes cliques in random graphs. [JBC+16] which defines the *suspiciousness* subgraph measure: for a subgraph S with $e(S)$ edges, the suspiciousness is $-\log P(Y = e(S))$, where Y has a Poisson distribution with mean equal to the expected mass of S under an Erdos-Renyi model. [vLDBSM16] proposes a subgraph interestingness measure. Their general framework evaluates how subjectively interesting a subgraph is relative to a user's prior beliefs.

Graph-Based Fraud Detection Building on singular value decomposition (SVD), latent factor models, and belief propagation (BP), these model the entire graph to find fraud. SpokEn [PSS+10] considered the "eigenspokes" pattern produced by pairs of eigenvectors of graphs, and was later generalized for fraud detection [JCB+14b]. FBox [SBGF14] builds on SVD but focuses on detecting attacks missed by spectral techniques. Several methods have used HITS [**?**]-like ideas to detect fraud in graphs [JCB+14a, GGMP04, CSYP12, GVK+12, WGD06]. BP has been used for fraud classification on eBay [PCWF07], and fraud detection [ACF13]. [SBGF14] performs adversarial analysis for spectral algorithms, showing that attacks of small enough scale will necessarily evade detection methods which rely on the top k SVD components.

A different direction for fraud detection focuses on local subgraphs, by analyzing the properties of egonets to detect fraud [CPV01, PAISM14]. CopyCatch [BXG+13] and GetTheScoop

[JCB$^+$14b] use local search heuristics to find relevant dense bipartite subgraphs. However, without guarantees on the search algorithm, the algorithms may not be robust to intelligent adversaries. Finding dense subgraphs has been an important focus of graph theory communities and has been studied from a wide array of perspectives [GTV11**?**]. [Cha00] finds subgraphs with large average degree, showing that subgraph average degree can be optimized with approximation guarantees. Variants have been proposed to efficiently find large, dense subgraphs [Tso15], with approximation guarantees.

5.2 Time Series

Time Series Forecasting Classical time-series forecasting methods include autoregression (AR)-based methods, including ARMA, ARIMA [BJRL15], seasonal ARIMA [BJRL15], and vector autoregression (VAR) [Ham94]. Exponential smoothing (ETS) models [Win60], including Holt-Winters [Win60] capture trends and seasonal patterns. Other methods include Kalman filtering [K$^+$60], Hidden Markov Models (HMMs) [LRBP09], and non-linear dynamical systems [MS16].

Temporal Fraud Detection There are a number of works on anomaly detection in multivariate time series [LH07, CTPK09, VS10, RRS00]. [BXG$^+$13] focuses on fraudulent temporal patterns in graphs, and [FCYJMT$^+$15] found suspicious inter-arrival times between events in social media. A couple of works address temporal patterns of reviews, e.g. [XWLY12] detects spam singleton reviews and [GGF14] detects time periods of unusual activity. A wide body of research has focused on understanding user behavior and especially rating behavior. In particular ratings have been studied by the recommendation systems community, with both frequentist [Kor08] and Bayesian models [SM08] demonstrating great success. Additionally some models have worked to take into account temporal features [Kor10], and others have captured the bimodal patterns in ratings data [BMFS14].

Change Detection Change detection methods aim to segment a time series into two or more regimes. Many approaches rely on a parametric statistical model (e.g. Gaussian): [CZ64, Lor71] allow the mean to change at each change point. This was extended to allow the Gaussian variance to change as well [BP98, KCG$^+$15]. This was extended to Poisson distributions for count data [Chi98, KCG$^+$15], and more general exponential family distributions [FMS14]. A number of nonparametric change detection methods have also been proposed. The Multivariate Rho (MRHO) [KQR16], Energy-Divisive (EDIV) [MJ14], and E-statistic (ECP) [MJ14] approaches use nonparametric goodness-of-fit measures as cost functions. Other methods include Kernel Change Point Detection (KCP) [DDD05], and neural networks [LLJ02]. FLOSS [GDY$^+$17] is a domain-agnostic online segmentation approach, but only for univariate data.

5.3 Dynamic Graphs

Dynamic Graph Change Detection Change detection methods for dynamic graphs, or graphs which change over time, have been proposed [DLJL09, SFPY07, PC15]. These include Bayesian methods [PC15], and distance-based methods, which define a distance metric on graphs: based on diameter [GKW06], node or edge weights [PDGM10, Pin05], connectivity [KVF13], or subgraphs [MBS13]. [AF10] finds change points in time-varying networks of agents, with edges representing interactions between them.

A number of methods consider graph scan statistics [MBR$^+$13, CN14, MSN13, RAGT14], which search for a highly anomalous area in static and temporal graphs. [CCV17] uses coloring to efficiently optimize graph scan statistics, and [CBVD18] incorporates uncertainty in the observed data. [SSR13] defines the Spatial Scan Statistic while [SRS16] defines the Graph Fourier Scan Statistic (GFSS), which quantify the spatial locality of a signal.

Dynamic Graph Anomaly Detection and Sensor Selection [AMF10] finds anomalous changes in graphs using an egonet-based approach, while [CHS12] uses a community-based approach. [MBR$^+$13] finds connected regions with high anomalousness. [APG$^+$14, SKZ$^+$15] detect large and/or transient communities using Minimum Description Length. Other partition-based [AZP11] and sketch-based [RHSS16] exist for anomaly detection.

For sensor selection, [LKG$^+$07] proposed CELF, for outbreak detection in networks, such as water distribution networks and blog data, also using a submodular objective function. For epidemics, [PSV02, CHBA03] consider targeted immunization, such as identifying high-degree [PSV02] or well-connected [CHBA03] nodes. A number of works consider the Optimal PMU Placement (OPP) problem [BH05], of optimally placing sensors in power grids, typically to make as many nodes as possible fully observable, or in some cases, minimizing mean-squared error. [BH05] shows that this problem is NP-complete. Hence, greedy [LNI11], convex relaxation [KGW12], integer program [DDGS08, Gou08], simulated annealing [BMBA93], and swarm-based [CVK08] approaches have been proposed.

Chapter 6

Timeline

Table 6.1 shows my expected timeline.

Table 6.1: Expected timeline.

Date	**Task**
October 2018	Thesis proposal
November 2018	Graphs (Section 2)
December 2018	Time series (Section 3)
January 2019	Dynamic graphs (Section 4)
January - April 2019	Interviewing
March - April 2019	Thesis writing
May 2019	Thesis defense

Chapter 7

Conclusion

In this thesis, we address the problem of anomaly detection in large-scale datasets. Our work makes contributions over three main thrusts:

- **Graphs** (Chapter 2) We develop a probabilistic score for how abnormal a subgraph is, and an algorithm for detecting dense subgraphs that is robust against adversarial manipulation.
- **Time Series** (Chapter 3) We perform anomaly detection in a categorical data setting arising from online systems, as well as more traditional real-valued sensor data. We also develop an online nonparametric anomaly detection approach that can be used on categorical, numeric or ordinal data.
- **Dynamic Graphs** (Chapter 4) We develop algorithms for detecting changes occured at a certain time which affects a connected subgraph of sensors. We then develop an algorithm for near-optimally selecting locations for sensor placement.

For reproducibility and the benefit of the community, we make the algorithms and datasets used available at `www.andrew.cmu.edu/user/bhooi/code`.

Bibliography

[AC09] Reid Andersen and Kumar Chellapilla. Finding dense subgraphs with size bounds. In *WAW*, 2009.

[ACF13] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. *ICWSM*, 13:2–11, 2013.

[AF10] Leman Akoglu and Christos Faloutsos. Event detection in time series of mobile communication graphs. In *Army science conference*, pages 77–79, 2010.

[AMF10] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, pages 410–421. Springer, 2010.

[And10] Reid Andersen. A local algorithm for finding dense subgraphs. *Transaction on Algorithms*, 6(4):60, 2010.

[ANMJZ12] Hélio Almeida, Dorgival Guedes Neto, Wagner Meira Jr, and Mohammed J Zaki. Towards a better quality metric for graph cluster evaluation. *Journal of Information and Data Management*, 3(3):378, 2012.

[APG$^+$14] Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E Papalexakis, and Danai Koutra. Com2: fast automatic discovery of temporal (âĂŸcometâĂŹ) communities. In *PAKDD*, pages 271–283. Springer, 2014.

[ARS02] James Abello, Mauricio GC Resende, and Sandra Sudarsky. Massive quasi-clique detection. In *Latin American symposium on theoretical informatics*, pages 598–612. Springer, 2002.

[AZP11] Charu C Aggarwal, Yuchen Zhao, and S Yu Philip. Outlier detection in graph streams. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 399–409. IEEE, 2011.

[BE76] Béla Bollobás and Paul Erdös. Cliques in random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 80, pages 419–427. Cambridge Univ Press, 1976.

[BGSLW93] Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson. A note on the prize collecting traveling salesman problem. *Mathematical programming*, 59(1-3):413–420, 1993.

[BH05] Dennis J Brueni and Lenwood S Heath. The pmu placement problem. *SIAM Journal on Discrete Mathematics*, 19(3):744–761, 2005.

[BJRL15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[BMBA93] TL Baldwin, L Mili, MB Boisen, and R Adapa. Power system observability with minimal phasor measurement placement. *IEEE Transactions on Power Systems*, 8(2):707–715, 1993.

[BMFS14] Alex Beutel, Kenton Murray, Christos Faloutsos, and Alexander J Smola. Cobafi: collaborative bayesian filtering. In *Proceedings of the 23rd international conference on World wide web*, pages 97–108. ACM, 2014.

[BP98] Jushan Bai and Pierre Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, pages 47–78, 1998.

[BXG$^+$13] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 119–130, 2013.

[CBVD18] Jose Cadena, Arinjoy Basak, Anil Vullikanti, and Xinwei Deng. Graph scan statistics with uncertainty, 2018.

[CCV17] Jose Cadena, Feng Chen, and Anil Vullikanti. Near-optimal and practical algorithms for graph scan statistics. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 624–632. SIAM, 2017.

[Cha00] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.

[CHBA03] Reuven Cohen, Shlomo Havlin, and Daniel Ben-Avraham. Efficient immunization strategies for computer networks and populations. *Physical review letters*, 91(24):247901, 2003.

[Chi98] Siddhartha Chib. Estimation and comparison of multiple change-point models. *Journal of econometrics*, 86(2):221–241, 1998.

[CHS12] Zhengzhang Chen, William Hendrix, and Nagiza F Samatova. Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems*, 39(1):59–85, 2012.

[CN14] Feng Chen and Daniel B Neill. 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*, pages 1166–1175. ACM, 2014.

[Coh08] Jonathan Cohen. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, 16, 2008.

[CPV01] Corinna Cortes, Daryl Pregibon, and Chris Volinsky. *Communities of interest.* Springer, 2001.

[CSYP12] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, 2012.

[CTPK09] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven A Klooster. Detection and characterization of anomalies in multivariate time series. In *SDM*, pages 413–424. SIAM, 2009.

[CVK08] Saikat Chakrabarti, Ganesh K Venayagamoorthy, and Elias Kyriakides. Pmu placement for power system observability using binary particle swarm optimization. In *Power Engineering Conference, 2008. AUPEC'08. Australasian Universities*, pages 1–5. IEEE, 2008.

[CZ64] Herman Chernoff and Shelemyahu Zacks. Estimating the current mean of a normal distribution which is subjected to changes in time. *The Annals of Mathematical Statistics*, 35(3):999–1018, 1964.

[DDD05] Frédéric Desobry, Manuel Davy, and Christian Doncarli. An online kernel change detection algorithm. *IEEE Transactions on Signal Processing*, 53(8):2961–2974, 2005.

[DDGS08] Devesh Dua, Sanjay Dambhare, Rajeev Kumar Gajbhiye, and SA Soman. Optimal multistage scheduling of pmu placement: An ilp approach. *IEEE Transactions on Power delivery*, 23(4):1812–1820, 2008.

[DLJL09] Dongsheng Duan, Yuhua Li, Yanan Jin, and Zhengding Lu. Community mining on dynamic weighted directed graphs. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 11–18. ACM, 2009.

[FCYJMT$^+$15] Alceu Ferraz Costa, Yuto Yamaguchi, Agma Juci Machado Traina, Caetano Traina Jr, and Christos Faloutsos. Rsc: Mining and modeling temporal activity in social media. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–278. ACM, 2015.

[FMS14] Klaus Frick, Axel Munk, and Hannes Sieling. Multiscale change point inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(3):495–580, 2014.

[GDY$^+$17] Shaghayegh Gharghabi, Yifei Ding, Chin-Chia Michael Yeh, Kaveh Kamgar, Liudmila Ulanova, and Eamonn Keogh. Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 117–126. IEEE, 2017.

[GGF14] Stephan Günnemann, Nikou Günnemann, and Christos Faloutsos. Detecting anomalies in dynamic rating data: A robust probabilistic model for rating evolution. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 841–850. ACM, 2014.

[GGMP04] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *VLDB Endowment*, pages 576–587, 2004.

[GKW06] Matthew E Gaston, Miro Kraetzl, and Walter D Wallis. Using graph diameter for change detection in dynamic networks. *Australasian Journal of Combinatorics*, 35:299, 2006.

[Gol84] Andrew V Goldberg. *Finding a maximum density subgraph.* Technical Report, 1984.

[Gou08] Bei Gou. Optimal placement of pmus by integer linear programming. *IEEE Transactions on Power Systems*, 23(3):1525–1526, 2008.

[GTV11] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 87–93. IEEE, 2011.

[GVK$^+$12] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. Understanding and combating link farming in the twitter social network. In *21st WWW*, pages 61–70. ACM, 2012.

[HAE$^+$18] Bryan Hooi, Leman Akoglu, Dhivya Eswaran, Amritanshu Pandey, Marko Jereminov, Larry Pileggi, and Christos Faloutsos. Changedar: Online localized change detection for sensor data on a graph. 2018.

[Ham94] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[HES$^+$18] Bryan Hooi, Dhivya Eswaran, Hyun Ah Song, Amritanshu Pandey, Marko Jereminov, Larry Pileggi, and Christos Faloutsos. Changedar: Online localized change detection for sensor data on a graph. 2018.

[HF18] Bryan Hooi and Christos Faloutsos. Branch and border: Partition-based change detection in multivariate time series. 2018.

[HSB$^+$16a] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. Birdnest: Bayesian inference for ratings-fraud detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 495–503. SIAM, 2016.

[HSB$^+$16b] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–904. ACM, 2016.

[HSLF18] Bryan Hooi, Kijung Shin, Hemank Lemba, and Christos Faloutsos. Telltail: a principled scoring function for dense subgraphs. 2018.

[HSP$^+$18] Bryan Hooi, Hyun Ah Song, Amritanshu Pandey, Marko Jereminov, Larry Pileggi, and Christos Faloutsos. Streamcast: Fast and online mining of power grid time sequences. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 531–539. SIAM, 2018.

[JBC$^+$16] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. Spotting suspicious behaviors in multimodal data: A general metric and algorithms. *TKDE*, 28(8):2187–2200, 2016.

[JCB$^+$14a] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Catchsync: catching synchronized behavior in large directed graphs. In *20th KDD*, pages 941–950. ACM, 2014.

[JCB$^+$14b] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Inferring strange behavior from connectivity pattern in social networks. In *Advances in Knowledge Discovery and Data Mining*, pages 126–138. Springer, 2014.

[JPS$^+$17] Marko Jereminov, Amritanshu Pandey, Hyun Ah Song, Bryan Hooi, Christos Faloutsos, and Larry Pileggi. Linear load model for robust power system analysis. In *IEEE PES Innovative Smart Grid Technologies*, page (submitted). IEEE, 2017.

[K$^+$60] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[KCG$^+$15] Stanley IM Ko, Terence TL Chong, Pulak Ghosh, et al. Dirichlet process hidden markov multiple change-point model. *Bayesian Analysis*, 10(2):275–296, 2015.

[KE11] Daniel Kahneman and Patrick Egan. *Thinking, fast and slow*, volume 1. Farrar, Straus and Giroux New York, 2011.

[KGW12] Vassilis Kekatos, Georgios B Giannakis, and Bruce Wollenberg. Optimal placement of phasor measurement units via convex relaxation. *IEEE Transactions on power systems*, 27(3):1521–1530, 2012.

[Kor08] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[Kor10] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[KQR16] Ivan Kojadinovic, Jean-François Quessy, and Tom Rohmer. Testing the constancy of spearmanâĂŹs rho in multivariate time series. *Annals of the Institute of Statistical Mathematics*, 68(5):929–954, 2016.

[KVF13] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. In *SDM*, pages 162–170. SIAM, 2013.

[LADW05] Lun Li, David Alderson, John C Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005.

[LCK$^+$10] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[LH07] Xiaolei Li and Jiawei Han. Mining approximate top-k subspace anomalies in multi-dimensional time-series data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 447–458. VLDB Endowment, 2007.

[LKG$^+$07] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks.

In *KDD*, pages 420–429. ACM, 2007.

[LLJ02] X Liu and RG Lathrop Jr. Urban change detection based on an artificial neural network. *International Journal of Remote Sensing*, 23(12):2513–2518, 2002.

[LNI11] Qiao Li, Rohit Negi, and Marija D Ilić. Phasor measurement units placement for power system state estimation: A greedy approach. In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–8. IEEE, 2011.

[Lor71] Gary Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, pages 1897–1908, 1971.

[LRBP09] Julie Letchner, Christopher Re, Magdalena Balazinska, and Matthai Philipose. Access methods for markovian streams. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 246–257. IEEE, 2009.

[MBR$^+$13] Misael Mongiovi, Petko Bogdanov, Razvan Ranca, Evangelos E Papalexakis, Christos Faloutsos, and Ambuj K Singh. Netspot: Spotting significant anomalous regions on dynamic networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 28–36. SIAM, 2013.

[MBS13] Misael Mongiovi, Petko Bogdanov, and Ambuj K Singh. Mining evolving network processes. In *ICDM*, pages 537–546. IEEE, 2013.

[MJ14] David S Matteson and Nicholas A James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 109(505):334–345, 2014.

[MS16] Yasuko Matsubara and Yasushi Sakurai. Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1045–1054. ACM, 2016.

[MSN13] Edward McFowland, Skyler Speakman, and Daniel B Neill. Fast generalized subset scan for anomalous pattern detection. *JMLR*, 14(1):1533–1561, 2013.

[PAISM14] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *20th KDD*, pages 1346–1355. ACM, 2014.

[PC15] Leto Peel and Aaron Clauset. Detecting change points in the large-scale structure of evolving networks. In *AAAI*, pages 2914–2920, 2015.

[PCWF07] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210. ACM, 2007.

[PDGM10] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1(1):19–30, 2010.

[PI75] James Pickands III. Statistical inference using extreme order statistics. *the Annals of Statistics*, pages 119–131, 1975.

[Pin05] Brandon Pincombe. Anomaly detection in time series of graphs using arma processes. *Asor Bulletin*, 24(4):2, 2005.

[PSS$^+$10] BA Prakash, M Seshadri, A Sridharan, S Machiraju, and C Faloutsos. Eigenspokes: Surprising patterns and community structure in large graphs. *PAKDD, 2010a*, 84, 2010.

[PSV02] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical Review E*, 65(3):036104, 2002.

[RAGT14] Polina Rozenshtein, Aris Anagnostopoulos, Aristides Gionis, and Nikolaj Tatti. Event detection in activity networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1176–1185. ACM, 2014.

[RHSS16] Stephen Ranshous, Steve Harenberg, Kshitij Sharma, and Nagiza F Samatova. A scalable approach for outlier detection in edge streams using sketch-based approximations. In *SDM*, pages 189–197. SIAM, 2016.

[RRS00] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, volume 29, pages 427–438. ACM, 2000.

[SBGF14] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. Spotting suspicious link behavior with fbox: an adversarial perspective. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 959–964. IEEE, 2014.

[SERF16] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. Corescope: Graph mining using k-core analysis - patterns, anomalies and algorithms. In *ICDM*, 2016.

[SF78] Stephen B Seidman and Brian L Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology*, 6(1):139–154, 1978.

[SFPY07] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM, 2007.

[SKZ$^+$15] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *KDD*, pages 1055–1064. ACM, 2015.

[SM08] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[SRS16] James Sharpnack, Alessandro Rinaldo, and Aarti Singh. Detecting anomalous activity on networks with the graph fourier scan statistic. *IEEE Transactions on Signal Processing*, 64(2):364–379, 2016.

[SSR13] James Sharpnack, Aarti Singh, and Alessandro Rinaldo. Changepoint detection over graphs with the spectral scan statistic. In *Artificial Intelligence and Statistics*, pages 545–553, 2013.

[TBG$^+$13] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *SIGKDD*, pages 104–112, 2013.

[Tso15] Charalampos Tsourakakis. The k-clique densest subgraph problem. In *24th WWW*, pages 1122–1132. International World Wide Web Conferences Steering Committee, 2015.

[vLDBSM16] Matthijs van Leeuwen, Tijl De Bie, Eirini Spyropoulou, and Cédric Mesnage. Subjective interestingness of subgraph patterns. *Machine Learning*, 105(1):41–75, 2016.

[VS10] Alireza Vahdatpour and Majid Sarrafzadeh. Unsupervised discovery of abnormal activity occurrences in multi-dimensional time series, with applications in wearable systems. In *SDM*, pages 641–652. SIAM, 2010.

[WGD06] Baoning Wu, Vinay Goel, and Brian D Davison. Propagating trust and distrust to demote web spam. *MTW*, 190, 2006.

[Win60] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[XWLY12] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 823–831. ACM, 2012.

[YH18] Hiroki Yanagisawa and Satoshi Hara. Discounted average degree density metric and new algorithms for the densest subgraph problem. *Networks*, 71(1):3–15, 2018.

[ZMST11] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems*, 26(1):12–19, 2011.