SpotLight: Detecting Anomalies in Streaming Graphs

Dhivya Eswaran[*]
Christos Faloutsos[*]
{deswaran,christos}@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Sudipto Guha
sudipto@amazon.com
Amazon
New York City, NY, USA

Nina Mishra
nmishra@amazon.com
Amazon
Palo Alto, CA, USA

ABSTRACT

How do we spot interesting events from e-mail or transportation logs? How can we detect port scan or denial of service attacks from IP-IP communication data? In general, given a sequence of weighted, directed or bipartite graphs, each summarizing a snapshot of activity in a time window, how can we spot anomalous graphs containing the sudden appearance or disappearance of large dense subgraphs (e.g., near bicliques) in near real-time using sublinear memory? To this end, we propose a randomized sketching-based approach called SpotLight, which guarantees that an anomalous graph is mapped 'far' away from 'normal' instances in the sketch space with high probability for appropriate choice of parameters. Extensive experiments on real-world datasets show that SpotLight (a) improves accuracy by at least 8.4% compared to prior approaches, (b) is fast and can process millions of edges within a few minutes, (c) scales linearly with the number of edges and sketching dimensions and (d) leads to interesting discoveries in practice.

CCS CONCEPTS

• **Information systems** → **Data stream mining**; • **Theory of computation** → *Graph algorithms analysis*;

KEYWORDS

Anomaly detection; streaming graphs; graph sketching

ACM Reference Format:
Dhivya Eswaran, Christos Faloutsos, Sudipto Guha, and Nina Mishra. 2018. SpotLight: Detecting Anomalies in Streaming Graphs. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3220040

1 INTRODUCTION

Time-evolving (or dynamic) weighted directed/bipartite graphs, where both nodes and edges are continuously added over time, are artifacts generated in many real-world contexts. Examples include transportation logs (w cabs travel from location s to location d), network communication logs (w packets sent by IP address s to IP address d), instant-messaging, phone call, e-mail logs (w messages/calls/emails from user s to user d), collaborative editing logs (w edits made by user s to page d) and so on.

[*]Work performed while at Amazon.

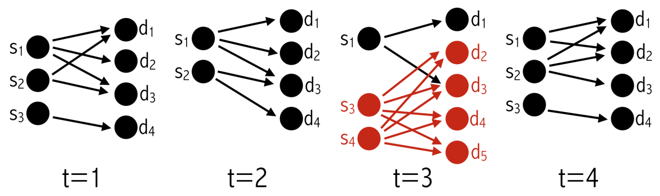


Figure 1: Sudden appearance of a dense subgraph at t=3.

We consider the problem of near real-time anomaly detection in such settings. Due to the fluid nature of what is considered 'normal', prior works typically focus on detecting specific anomalous changes to the graph, e.g., bridge edges [22, 26], hotspot nodes [31], changes to community structure [27, 28], graph metrics [8, 10], etc. In this work, we focus on detecting anomalies involving the *sudden appearance or disappearance of a large dense directed subgraphs* (near bicliques), which is useful in numerous applications: detecting attacks (port scan, denial of service) in network communication logs, interesting/fraudulent behavior creating spikes of activity in user-user communication logs (scammers who operate fast and in bulk), important events (holidays, large delays) creating abnormal traffic in/out flow to certain locations, etc. We are able to discover several of the above phenomena in real-world data (e.g., Fig. 12).

We highlight two important aspects of the above definition. The (dis)appearance of a large dense subgraph is anomalous only if it is *sudden*, i.e., it has not been observed before or is not part of a slow evolution (e.g., steadily growing communities). Similarly, the sudden (dis)appearance of a large number of edges is anomalous only if the edges form a *dense* subgraph (the so-called *lockstep* behavior indicating fraud [5]). Fig. 1 illustrates this. In the evolution of a bipartite graph, e.g., user edits page, an anomalous dense directed subgraph appears at t=3, indicating a possible edit-war between users s_3 and s_4 w.r.t. pages d_2, d_3, d_4, d_5. In contrast, the appearance of subgraph $\{s_1, s_2\} \to \{d_1, d_2, d_3\}$ at t=4 is not anomalous, since it has already been (partially) observed at t=1, 2.

The temporal aspect, i.e., near real-time detection, is crucial for our problem. The value of a newfound surge of ridership requests or network attack lies in the moment, not one week later. Moreover, given that nodes and edges are added over time, we seek solutions that can operate in sublinear memory, without storing a counter for each edge/node. The problem we set out to solve is:

Problem 1. ***Given*** *a stream of weighted, directed/ bipartite graphs,* $\{\mathcal{G}_1, \mathcal{G}_2, \ldots\}$, ***detect in near real-time*** *whether* $\mathcal{G}_t$ *contains a sudden (dis)appearance of a* large dense *directed subgraph* ***using sublinear memory.***

The technical challenge in detecting the sudden (dis)appearance of a large dense directed subgraph is computational. New edges and nodes are continuously arriving and we have limited time and space to process the changes. The approach that we take is to design a short summary or sketch of the graph that both reveals newly found anomalies and can be quickly updated and maintained on a high-speed moving data stream.

Concretely, our contributions are: **(a) Algorithm (Sec. 4):** We propose SpotLight, a simple randomized sketching-based approach to solve Problem. 1. **(b) Guarantees (Sec. 5):** We prove that SpotLight is *focus-aware* in expectation, i.e., flags focused addition or deletion of edges as more anomalous than dispersed changes of the same magnitude (Thm. 5.2) and maps anomalous graphs 'far' away from 'normal' instances in the sketch space *with high probability* for appropriate choice of parameters (Thm. 5.3). **(c) Effectiveness (Sec. 6):** Extensive experiments on real-world data show that SpotLight outperforms prior approaches in terms of precision and recall, is fast and scalable and leads to interesting discoveries.

2 RELATED WORK

Anomaly detection in static graphs is well-studied (for survey, see [4]). Unsupervised methods rely on node-level features [3], spectral decomposition [21], finding dense subgraphs signifying fraud [5, 11], etc. In the presence of limited supervision, belief propagation is known to work well [7].

Anomaly detection in time-evolving graphs can be reviewed under the following categories (for survey, see [23]).

(i) Approaches comparing consecutive snapshots [14, 26]: The traditional approach is to compare adjacent graphs $(\mathcal{G}_t, \mathcal{G}_{t+1})$ via a similarity function based on, e.g., belief propagation [14], random walks [26], etc., They do not consider evolutionary/periodic trends.

(ii) Dense subgraph detection based approaches [13, 25]: These techniques model dynamic graphs as node×node×time tensors and aim to approximately identify the top-k densest subblocks, e.g., persistent dense subgraphs. In contrast, we aim to detect only the *sudden* appearance of dense subgraphs in *near real-time.*

(iii) Graph decomposition/partitioning based approaches [27, 28]: These methods store a summary of the graph structure based on tensor decomposition [28] or minimum description language [27] and identify change points as anomalies. Their primary focus is on the computationally hard problem of graph modeling.

(iv) Anomalous edge detection approaches [2, 17, 22]: The first two methods score the likelihood of an edge based on the community structure [2], prior occurrence preferential attachment and homophily information [22]. By scoring edges independent of each other, these methods miss complex structural (e.g., dense subgraph) anomalies. They also cannot detect edges which are expected but do not occur. [17] is closely related, but is applicable when only multiple heterogeneous graphs are evolving simultaneously.

(v) Others: [10] offers a suite graph metrics to perform anomaly detection at multiple temporal and spatial granularities. [12] detects anomalous nodes using their activity vectors from principle component analysis (PCA). [31] also uses PCA, but to detect anomalous nodes (hotspots). [8] proposes density-consistent statistics to compare graphs having significantly different edge counts.

A qualitative comparison is provided in Table 1.

Property	**[14], [26]**	**[28],[27]**	**[31]**	**[12]**	**[10]**	**[22]**	**This work**
Directed/bipartite graphs		✔		?	?		✔
Weighted/multi edges	✔	✔	✔	✔	✔	✔	✔
Sublinear memory						✔	✔
Theoretical guarantee							✔

Table 1: Qualitative comparison with prior work on anomaly detection in streaming graphs.

Randomized graph streaming algorithms for testing connectivity and bipartiteness, constructing sparsifiers and spanners, approximating the densest subgraph etc. in the semi-streaming model (in $O(n \text{ polylog } n)$ space where n is the number of nodes) are popular within the theory community [18, 19]. However, they do not address graph *anomaly detection* using *sublinear memory.*

Randomized algorithms for anomaly detection: Perhaps, the first known randomized anomaly detector is Isolation Forests [16] for static multi-dimensional data. Due to its empirical success, randomized algorithms for streaming multi-dimensional data streams are recently gaining traction [9, 20, 29]. In this work, we investigate a randomized algorithm for the streaming *graph* setting.

3 PRELIMINARIES

In this section, we introduce our streaming model and formalize how to detect the sudden (dis)appearance of large dense subgraphs.

Streaming model. Let $\mathfrak{G} = \{\mathcal{G}_t\}_{t=1}^{\infty}$ be a graph stream. Each graph $\mathcal{G}_t$ is a tuple $(\mathcal{S}_t, \mathcal{D}_t, \mathcal{E}_t)$ where $\mathcal{S}_t$ and $\mathcal{D}_t$ are the possibly time-evolving sets of source and destination nodes respectively and each edge (s, d, w) in the edge set $\mathcal{E}_t$ originates from a *source* $s \in \mathcal{S}_t$, ends at a *destination* $d \in \mathcal{D}_t$ and carries a *weight* $w \in \mathbb{R}^+$ (w=0 is equivalent to the absence of an edge). We assume each node (source or destination) has a unique identifier that is fixed over time, i.e., the *node-correspondence* across graphs is known. Let $\mathbf{A}_t = [A_{t,sd}]$ be the adjacency of $\mathcal{G}_t$ where each $A_{t,sd}$ denotes the sum of weight of edges connecting a source s to a destination d in graph $\mathcal{G}_t$. While there are other ways of aggregating weights, this is the most natural in the applications we consider (see Sec. 1) .

The above model allows us to represent a flexible range of graphs: (i) weighted or unweighted (by letting $A_{t,sd} = 1 \; \forall \; s, d$), (ii) bipartite or unipartite (by allowing $\mathcal{S}_t$ and $\mathcal{D}_t$ to overlap) and (iii) directed or undirected (by constraining $A_{t,sd} = A_{t,ds}$) when $s \neq d$).

Problem Description. Given a graph $\mathcal{G}$ with adjacency $\mathbf{A}$, let $\mathcal{G}(\mathcal{S}', \mathcal{D}')$ denote the directed subgraph induced by the source set $\mathcal{S}'$ and the destination set $\mathcal{D}'$. Its density $\rho(\mathcal{G}(\mathcal{S}', \mathcal{D}'))$ can be defined in several ways, e.g., $\sum_{s \in \mathcal{S}', d \in \mathcal{D}'} A_{sd} / |\mathcal{S}'| \, |\mathcal{D}'|$ – the higher the total weight of edges in it, the greater its density [6].

In a nutshell, a graph $\mathcal{G}_t$ is said to be anomalous – i.e., contain a sudden appearance or disappearance of a dense directed subgraph – if there is a *large* directed subgraph which shows a *significant* change in density compared to the past graphs $\{\mathcal{G}_{t-1}, \mathcal{G}_{t-2}, \ldots\}$. For example, in Fig. 1, letting $\mathcal{S}' = \{s_3, s_4\}$ and $\mathcal{D}' = \{d_2, d_3, d_4, d_5\}$, the subgraph $\mathcal{G}_3(\mathcal{S}', \mathcal{D}')$ has high density (=1) but $\mathcal{G}_1(\mathcal{S}', \mathcal{D}')$ and $\mathcal{G}_2(\mathcal{S}', \mathcal{D}')$ have low densities, 0.125 and 0 respectively. Hence $\mathcal{G}_3$ is an anomaly. The next section presents the proposed method to identify such anomalies.

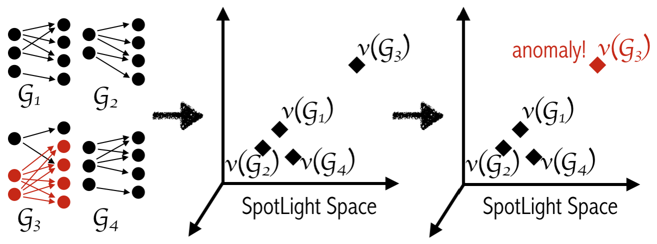


Figure 2: Overview of SpotLight

4 PROPOSED METHOD

The proposed method, called SpotLight, works in two main steps as shown in Alg. 1. First, it extracts a K-dimensional (we show how to choose K in Sec. 5) SpotLight sketch $\mathbf{v}(\mathcal{G})$ for every $\mathcal{G}$, such that graphs containing the sudden (dis)appearance of large dense subgraphs are 'far' from 'normal' graphs in the sketch space (line 4). Second, it exploits the distance gap in the sketch space to detect graphs yielding anomalous sketches as anomalous graphs (line 5). A schematic is given in Fig. 2. We next elaborate on these two steps in greater detail.

4.1 SpotLight graph sketching

A natural way to sketch a graph is by enumerating the total edge weight of each directed subgraph $\mathcal{G}(\mathcal{S}', \mathcal{D}')$ for sufficiently large source and destination sets $\mathcal{S}', \mathcal{D}'$. However, this sketch has exponential number of dimensions and is infeasible to compute or store. Hence, we propose to compose a sketch containing total edge weights of K specific directed subgraphs (called *query subgraphs* henceforth) chosen independently and uniformly at random, according to node sampling probabilities, p for sources and q for destinations. This leads to (K, p, q)-SpotLight graph sketching.

Conceptually, SpotLight sketching first chooses K query subgraphs $\{(\mathcal{S}'_k, \mathcal{D}'_k)\}_{k=1}^{K}$ by sampling each source (or destination) into each $\mathcal{S}'_k$ (resp. $\mathcal{D}'_k$) with probability p (resp. q). This choice is made only once per source or destination (the first time it is seen) and is fixed throughout the graph stream. Next, for every graph $\mathcal{G}$, its sketch $\mathbf{v}(\mathcal{G}) \in \mathbb{R}^K$ is computed as $v_k(\mathcal{G}) = \sum_{s \in \mathcal{S}'_k, d \in \mathcal{D}'_k} A_{sd} = total\_edge\_weight(\mathcal{G}(\mathcal{S}'_k, \mathcal{D}'_k))$. For example, in Fig. 3 showing a graph $\mathcal{G}$ with unit-weight edges, there are three edges belonging to the first query subgraph (red), one to the second (green) and none to the third (blue). Hence, its sketch is $\mathbf{v}(\mathcal{G}) = (3, 1, 0)$.

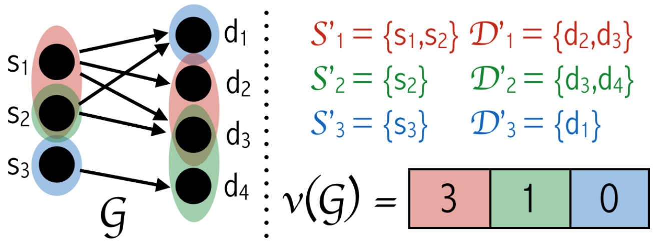

s1
s2
s3
d1
d2
d3
d4
$\mathcal{G}$
$\mathcal{S}'_1 = \{s_1, s_2\}$
$\mathcal{D}'_1 = \{d_2, d_3\}$
$\mathcal{S}'_2 = \{s_2\}$
$\mathcal{D}'_2 = \{d_3, d_4\}$
$\mathcal{S}'_3 = \{s_3\}$
$\mathcal{D}'_3 = \{d_1\}$
$v(\mathcal{G}) =$
3
1
0

Figure 3: A $(K{=}3, p{=}0.5, q{=}0.33)$-SpotLight sketch $\mathbf{v}(\mathcal{G})$ of a graph $\mathcal{G}$ with unit-weight edges. Each sketch dimension $v_k(\mathcal{G})$ is the total weight of edges going from a random set of sources $\mathcal{S}'_k$ and to a random set of destinations $\mathcal{D}'_k$.

Algorithm 1 SpotLight graph stream anomaly detection

```
Input: a stream $\mathfrak{G}$ of weighted directed/bipartite graphs
Parameters: sketch dimensionality $K$, source sampling probability $p$, destination sampling probability $q$
Output: a stream of anomaly scores
1: procedure SpotLight($\mathfrak{G}, K, p, q$)
2:     Initialize($K, p, q$)
3:     for graph $\mathcal{G} \in \mathfrak{G}$ do
4:         v ← Sketch($\mathcal{G}$)
5:         yield AnomalyScore(v)
6: procedure Initialize($K, p, q$)
7:     for $k = 1, \ldots, K$ do
8:         Pick source hash $h_k : \mathcal{S} \rightarrow \{1, \ldots, \lfloor 1/p \rfloor\}$ and destination hash $h'_k : \mathcal{D} \rightarrow \{1, \ldots, \lfloor 1/q \rfloor\}$ independently at random.
9: procedure Sketch($\mathcal{G}$)
10:    v ← $\mathbf{0}_K$
11:    for edge $e = (s, d, w)$ in graph $\mathcal{G}$ do
12:        for $k = 1, \ldots, K$ do
13:            if $h_k(s) == 1$ and $h'_k(d) == 1$ then
14:                $v_k \leftarrow v_k + w$
15:    return v
```

An efficient implementation of SpotLight sketching using hashing is given in Alg. 1. The hash functions ensure that the node to query subgraph mapping remains fixed over time without explicitly storing it. The choice of the first hash bucket in line 13 is arbitrary; one can pick any value within the suitable range. Observe how this algorithm is able to seamlessly process old and new nodes alike.

SpotLight sketching can be thought of in two alternative ways. First, it can be regarded as a memory-limited and non-deterministic generalization of two common used graph features – nodal degree $(K{=}|\mathcal{S}|, p{=}1/|\mathcal{S}|, q{=}1$ or $K{=}|\mathcal{D}|, p{=}1, q{=}1/|\mathcal{D}|)$ and total edge weight $(K{=}p{=}q{=}1)$. Second, and more interestingly, each sketch dimension can be considered as a spotlight which illuminates and allows for monitoring a region of the graph (i.e., its query subgraph). The central idea is that *the (dis)appearance of a large and dense subgraph would be brought to light by at least one of these spotlights, provided there are enough of them and each one is fine-grained, illuminating a small enough region of the graph.* In Sec. 5, we prove high probability guarantees of exactly this nature.

4.2 Anomaly detection in the SpotLight space

Exploiting the distance gap between the anomalous graphs containing the sudden (dis)appearance of large dense subgraphs and 'normal' instances in the SpotLight (sketch) space, we may now employ any off-the-shelf data stream anomaly detector (e.g., [9, 20, 29]) to carry out AnomalyScore procedure call (line 5 of Alg. 1). These techniques require sublinear memory and output an anomaly score for every data point (i.e., SpotLight graph sketch) in the stream.

5 THEORETICAL ANALYSIS

This section presents the distance guarantees offered by SpotLight sketch space and also analysis of running time and memory.

5.1 Guarantees for SpotLight sketches

How do we theoretically analyze the distance between graphs in the SpotLight space, even though the sketching algorithm is randomized? What properties should this distance function obey? How do we choose the sketching parameters so that anomalous graphs lie 'far' from 'normal' instances with high probability in the SpotLight space? These are the questions we set out to answer.

In the rest of this section, $\mathcal{G}$ is always an arbitrary weighted directed/bipartite graph on N_s sources and N_s destinations. Adding unit-weight edges to $\mathcal{G}$ increments corresponding edge weights by one, even if these edges already existed. $\mathbf{v}(\cdot)$ represents the (K, p, q)-SpotLight sketch. For simplicity, we let N_s=N_d=N and p=q. Also, without loss of generality, we consider only the appearance of dense subgraphs (disappearance can be argued in a similar way).

We begin by defining SL-distance (SL for SpotLight) between graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ in the SpotLight space as a *deterministic* function of $\mathcal{G}_1, \mathcal{G}_2$ and the sketching parameters K, p, q.

Definition 1 (SL-Distance). *The SL-distance between graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ is the expected squared Euclidean distance between their SpotLight sketches, i.e., $\bar{d}(\mathcal{G}_1, \mathcal{G}_2) = \mathbb{E}\left[||\mathbf{v}(\mathcal{G}_1) - \mathbf{v}(\mathcal{G}_2)||_2^2\right]$, where the expectation is taken over the random coin tosses of the sketching algorithm*[1].

We devote the rest of this section to show (i) that SL-distance is *focus-aware*, a desirable property for anomaly detection and (ii) how to set sketching parameters so that 'anomalous' graphs lie far from 'normal' ones according to SL-distance. All proofs are given in the appendix.

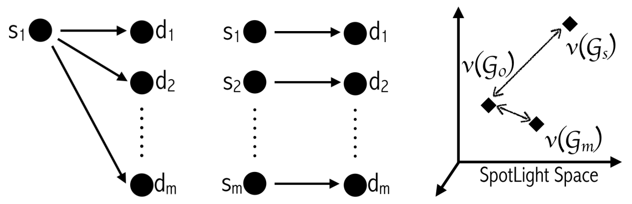
5.1.1 Focus-awareness. Many highly dynamic settings, e.g., IP-IP communication logs, present bursty traffic leading to a high variance in the total edge weight. Thus, it becomes easy for a sudden appearance of dense subgraph, e.g., denial of service attack, to evade detection, unless the distance function used has the so-called *focus-awareness* property: 'random [dispersed] changes in graphs are less important [anomalous] than targeted [focused] changes of the same extent' [14]. In this section, we show that SL-distance has this desirable property. Consider,

Example 1 (Star vs. Matching). *Add an out-star graph (Fig. 4a) of m unit-weight edges (focused change) to $\mathcal{G}$ to obtain $\mathcal{G}_S$. Add a matching graph (Fig. 4b) of m edges (dispersed change) to $\mathcal{G}$ to create $\mathcal{G}_M$. Intuitively, the appearance of a dense star subgraph is more anomalous (e.g., potential port scan attack/ hotspot in road traffic) and accordingly, we desire $\bar{d}(\mathcal{G}, \mathcal{G}_S) > \bar{d}(\mathcal{G}, \mathcal{G}_M)$. See Fig. 4c.*

We show that SL-distance not only satisfies the condition above, but even the distance gap increases with the number of edges m and sketch dimensionality K. That is, $\mathcal{G}_S$ is increasingly more anomalous than $\mathcal{G}_M$ as m grows. See Lem. 5.1.

Lemma 5.1 (Star vs. matching). *Suppose $\mathcal{G}$, $\mathcal{G}_S$ and $\mathcal{G}_M$ are as defined in Ex. 1, with (K, p, q)-SpotLight sketches $\mathbf{v}(\cdot) \in \mathbb{R}^K$ and let $0 < p, q < 1$. Then, $\bar{d}(\mathcal{G}_S, \mathcal{G}) > \bar{d}(\mathcal{G}_M, \mathcal{G}) + O\left(Km^2\right)$.*

The edge addition process in Ex. 1 was deterministic, in the sense that the relative position of added edges was fixed. We now consider the more general case where m edges are added uniformly at random (i.e., non-deterministically) in regions of different sizes. Thm. 5.2 shows that the smaller the region in which edges are added, the farther away the final graph lies from the initial graph in the SpotLight space (in expectation). In other words, the more focused the edge addition, the more anomalous the final graph is expected to be in the SpotLight-space.



(a) Star graph (b) Matching graph (c) Ideal sketches

Figure 4: Focus-awareness: Addition of dense star graph is more anomalous than that of the sparse matching graph.

Theorem 5.2 (Focus-Awareness). *Consider the distribution of graphs $\mathcal{F}'$ obtained by adding m unit-weight edges (in expectation) to any $n'{\times}n'$ region of $\mathcal{G}$ by sampling each of the n'^2 possible edges with probability m/n'^2. Let $\mathcal{F}''$ be another distribution over graph obtained by adding edges in a similar manner to any $n''{\times}n''$ region. Then,*

$$n'' < n' \implies \mathbb{E}_{\mathcal{G}''\sim\mathcal{F}''}\left[\bar{d}(\mathcal{G}, \mathcal{G}'')\right] > \mathbb{E}_{\mathcal{G}'\sim\mathcal{F}'}\left[\bar{d}(\mathcal{G}, \mathcal{G}')\right] \quad (1)$$

Thm. 5.2 guarantees a separation in the expected SL-distance, (the expectation is taken over the random coin tosses of the edge addition process), which is a *necessary* condition for anomaly detection to work. It is not sufficient, however: in order to detect $\mathcal{F}''$ as anomalies in the SpotLight space, a *large* distance gap with *high* probability is crucial. Sec. 5.1.2 addresses precisely this.

5.1.2 Criterion for anomaly detection. To show that anomalous graphs are mapped *far* from *normal* instances in the SpotLight space, we need formal definitions for (i) what 'far' means in the sketch space and (ii) what class of 'normal' graphs to use as a control group. These are provided in Def. 2 and Def. 3 respectively.

Definition 2 (ϵ-SL-Farness). *If $\bar{d}(\mathcal{G}_1, \mathcal{G}) > \bar{d}(\mathcal{G}_2, \mathcal{G}) + \epsilon$, we say that $\mathcal{G}_1$ is ϵ-SL-far from $\mathcal{G}$ compared to $\mathcal{G}_2$.*

Definition 3 (Erdős-Rényi Control Group). *Let $\mathcal{G}$ be a graph on N sources and N destinations. An Erdős-Rényi (ER) control group $\mathcal{F}_{ER}(\mathcal{G}, m)$ is defined as a distribution of graphs, where each instance $\mathcal{G}_{ER}$ is obtained by adding m unit-weight edges (in expectation) uniformly throughout the graph by sampling each of the N^2 possible edges independently with probability m/N^2.*

The choice of ER control group is motivated by focus-awareness: we wish to distinguish the addition of a dense subgraph of m edges in *any focused* part of the graph from a case where the same m edges are added uniformly at random throughout the graph. Thm. 5.3 asserts this is indeed the case: when sketching parameters are chosen appropriately, it is possible to achieve an ϵ-separation between the anomalous and normal graphs with high probability.

Theorem 5.3 (Anomaly Detection Criterion). *Add n^2 unit-weight edges in any $n{\times}n$ region to get $\mathcal{G}_{BC}$ (BC for BiClique). Let $1 \ll n^2 \ll N^2$ and $p = q < 0.5$. Then, $\mathcal{G}_{BC}$ is ϵ-SL-far from $\mathcal{G}$*

[1]$\bar{d}(\cdot, \cdot)$ is not a metric, but it obeys a relaxed triangle inequality.

compared to a $\mathcal{G}_{ER}$ drawn from $\mathcal{F}_{ER}(\mathcal{G}, n^2)$ with high probability $1-\delta$, i.e.,

$$\mathbf{Pr}_{\mathcal{G}_{ER}\sim\mathcal{F}_{ER}(\mathcal{G},n^2)}\left[\bar{d}(\mathcal{G},\mathcal{G}_{BC}) - \bar{d}(\mathcal{G},\mathcal{G}_{ER}) \geq \epsilon\right] \geq 1-\delta \qquad (2)$$

where δ is the false positive rate on the ER control group, provided:

$$K > \frac{(1+p^2n^2)^2}{4p^2n^2\delta} + \frac{\epsilon}{p^3n^3} \qquad (3)$$

Observe from Eq. (3) that more sketch dimensions are required if ϵ is high or δ is low which is intuitive: the higher the separation needed between the anomaly and the control group or the lower the permitted false positive rate on the control group, the more dimensions we need. Another subtle point to note here is that Thm. 5.3 guarantees an isolation of anomalies in the sketch space, *without knowing a priori which $n\times n$ region contains the dense subgraph* – this is crucial because, in practice, anomalous dense subgraphs can appear (or disappear) in any region. Further, Thm. 5.3 also guides us in choosing parameters, as stated below.

Corollary 5.4 (Optimal Sketching Parameters). *From Eq. (3), the optimal value of p requiring the least sketching dimensionality is obtained by solving $n^5p_*^5 - np_* = 6\epsilon\delta$. When ϵ=0, this reduces to p_*=$1/n$ i.e., sample exactly one* added *edge in expectation. Accordingly, we require $K_* > 1/\delta$.*

For example, with K=50, p=q=0.2, we may detect the addition of n=5 biclique as an anomaly with ϵ=0 separation by incurring at most δ=2% false positive rate on the ER control group.

5.2 Running time and memory analysis

SpotLight obeys the sublinear memory and linear time constraints of Problem. 1, as stated below.

Lemma 5.5 (Linear Running Time). *SpotLight takes $O(|\mathcal{E}|\cdot K)$ time to process each $\mathcal{G} = (\mathcal{S}, \mathcal{D}, \mathcal{E})$ in the stream.*

Lemma 5.6 (Sublinear Memory Requirement). *SpotLight takes $O(\log N_s + \log N_d + K)$ to process each graph in a stream having N_s sources and N_d destinations.*

SpotLight sketching runs in $O(|\mathcal{E}|\cdot K)$ running time due to the loops in lines 11-12 (Alg. 1), since the other steps require constant time. The $O(\log N_s + \log N_d)$ space is a lower bound on memory requirements, since each edge (including source and destination identifiers) needs to be read (one by one). An additional $O(K)$ space is needed to store the sketch. Anomaly detection in SpotLight space takes $O(K)$ time and sublinear space, e.g., using [9].

6 EXPERIMENTS

We empirically evaluate the proposed method on datasets where the anomalies are verifiable and interpretable. We begin with the details of datasets and experimental setup.

6.1 Datasets

We shortlist three real-world publicly available time-evolving graph datasets, where the anomalies can be verified by comparing to manual annotations or by correlating with real-world events:

Darpa dataset [15] contains $4.5M$ IP-IP communications taking place between 9484 source IPs and 23398 destination IPs over $87.7K$ time steps (minutes). Each communication is a directed edge *(srcIP, dstIP, 1, time)*. We obtain a stream of 1463 graphs by aggregating edges occurring in every hourly duration. The dataset contains 89 known network attacks – large or stealthy – e.g., `portsweep`, `ipsweep`, `mscan` and `snmpgetattack`. Most attacks were large (> 100 edges), but were targeted at and/or engineered from a few hosts and occurred in single/multiple bursts of time – thus, leading to the *sudden (dis)appearance of large dense subgraphs* that we aim to detect. Using the furnished ground truth (attack/not) for each edge, we label a graph as anomalous if it contains at least 50 attack edges.

Enron dataset [24] contains $\sim 50K$ emails exchanged among 151 employees of the energy company over a 3 year period surrounding the famous Enron scandal. Each email is a directed edge *(sender, receiver, 1, timestamp)*. We derive a stream of 1139 graphs by treating each day as its own graph. As ground truth is not directly available, we verify the detected anomalies by correlating with the major events of the scandal.

NycTaxi dataset [1] contains taxi ridership data during a 3-month period (Nov 2015–Jan 2016) obtained from New York City (NYC) Taxi Commission. Each taxi trip is furnished with pick-up (PU)/drop-off (DO) times and (lon, lat) coordinates of PU/DO locations, which we process as follows. We manually click on the centers of 57 geographically or conceptually distinguishable NYC *zones* based on common knowledge – including parks, airports, stadiums, bridges, residential neighborhoods, islands – on a map and note their (lon, lat) coordinates. Every PU/DO location is then assigned to the nearest *zone*. Thus, a directed edge *(srcZone, dstZone, 1, timestamp)* is created for each taxi trip. These are further aggregated into 2208 graphs, each containing trips that took place in a given hourly duration. We verify the detected anomalies by correlating with important occasions – holidays, events, unusual weather conditions – which affect the normal rhythm of road traffic.

6.2 Experimental Setup

We implement SpotLight (abbreviated as SL henceforth) in Python and run experiments on MacOS with 2.7 GHz Intel Core i5 processor and 16 GB main memory. By default, we use K=50 sketch dimensions and p=q=0.2 source/destination sampling probabilities. Mapping to Thm. 5.3, this corresponds to detecting a n=5 biclique (or more) as an anomaly w.r.t. the control group by incurring less than δ=2% false positives. This also ensures all edges are covered twice in expectation. For the anomalous sketch detection step, we use the state-of-the-art Robust Random Cut Forests (RRCF) [9] with 50 trees and 256 samples (unless specified otherwise).

Baselines: We compare SpotLight to the following three baselines on the labeled Darpa dataset: **(a) EdgeWeight (EW):** We consider a vanilla version of SL by setting K=p=q=1, i.e., sketching each graph using a single coarse-grained feature, namely, its total weight of edges. Observe that EW tends to miss 'small' anomalies which do not alter the total edge weight significantly compared to usual. **(b) RHSS [22]**, abbreviated based on the last names of authors, processes each edge e in the stream individually, outputting a likelihood score $\ell(e)$. We compute the likelihood of a graph $\mathcal{G} = (\mathcal{S}, \mathcal{D}, \mathcal{E})$ as the geometric mean of the per-edge likelihoods (similar to [2]): $\ell(\mathcal{G}) = (\prod_{e=(s,d,w)\in\mathcal{E}} \ell(e)^w)^{1/W}$ where W is the total edge weight. Finally, to reflect the intuition that a more likely

Method	precision@ 100	200	300	400	recall@ 100	200	300	400
Ideal	*1.0*	*1.0*	*0.96*	*0.72*	*0.35*	*0.69*	*1.0*	*1.0*
SL	**0.96**	**0.79**	**0.64**	**0.57**	**0.34**	**0.55**	**0.67**	**0.80**
EW	0.86	0.54	0.47	0.46	0.30	0.38	0.49	0.65
RHSS	0.31	0.28	0.32	0.36	0.11	0.19	0.33	0.50
STA	0.23	0.16	0.19	0.24	0.08	0.11	0.20	0.34

Table 2: SpotLight (SL) achieves better precision and recall than baselines (EW, RHSS, STA). Bold indicates the highest value in each column (excluding ideal). Underline shows significant differences (p-value ≤ 0.01) w.r.t. baselines according to a two-sided micro-sign test [30].

graph is less anomalous, we use $anomaly\_score(\mathcal{G}) = -\log \ell(\mathcal{G})$. We implement RHSS in Python without using the sketching-based approximation[2]. **(c) STA [28]** scores the anomalousness of each graph as the error incurred in reconstructing it based on a streaming graph decomposition. We use 50 as the rank of decomposition.

Evaluation Metrics: Each method above outputs an anomaly score (higher is anomalous) per graph. Sorting these in descending order, we compute the number of anomalies caught $TP(k)$ (true positives) among the top k most anomalous graphs, for every k. If the overall number of anomalies is N, we compute $precision@k = TP(k)/k$ and $recall@k = TP(k)/N$. We also summarize the overall accuracy using the AUC (Area Under ROC Curve) score. Recall that *precision@k*, *recall@k* and AUC lie in $[0, 1]$ and a higher value is better. In addition, we note the running time of all methods, averaged over five runs.

Experimental Design: Our experiments are designed to answer the following questions: **[Q1] Accuracy**: How well is SpotLight able to spot anomalies compared to baselines? What is the trade-off with respect to running time? How does the performance vary with parameters? **[Q2] Scalability**: How does the running time scale with the number of edges in the stream and sketch dimensions K? **[Q3] Discoveries**: Does SpotLight lead to interesting discoveries on real world data? We now present our findings.

Q1. Accuracy

Table 2, Fig. 5 and Fig. 6 compare the precision, recall, accuracy (AUC) and running time of SL with baselines on the labeled Darpa dataset. Fig. 7 shows the variation of accuracy with parameters. As SL and EW are initialized based on the first 256 graphs, performance is reported on the subsequent $1463 - 256 = 1207$ graphs, containing 288 ground truth anomalies (23.8% of total).

Precision and recall: Table 2 gives the precision and recall at cut-off ranks $k \in \{100, 200, 300, 400\}$. Ideal values are computed based on an oracle which scores the ground truth anomalies higher than all non-anomalies. We see that SL consistently outperforms all baselines achieving $11-46\%$ (statistically significant) improvements. Further, a plot of precision vs. recall for all methods, shown in Fig. 5, reveals that SL's curve (blue) lies completely above those of all baselines, achieving higher precision for every recall value. Thus, the performance gain of SL generalizes to all cut-off ranks (k).

[2]We also tried computing the anomaly score as the negative average of the per-edge likelihoods and obtained similar results.

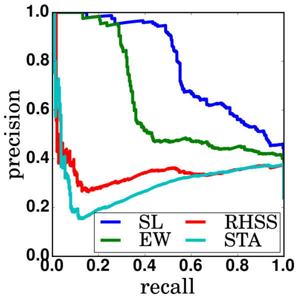


Figure 5: SL outperforms baselines in terms of precision and recall.

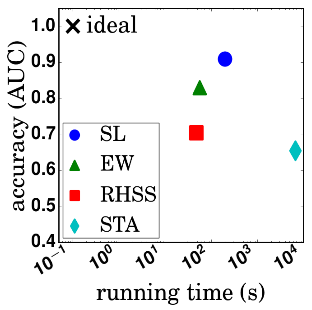
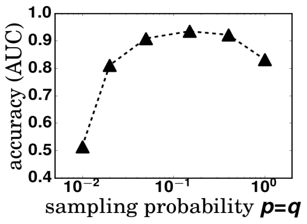
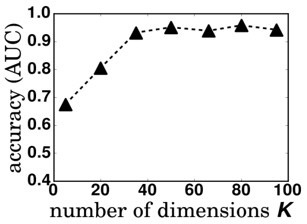


Figure 6: Accuracy-running time trade-off offered by SL.



(a)



(b)

Figure 7: Variation of accuracy with (a) $p = q$ for $K = 10$ and (b) with K for $p = q = 0.1$.

Accuracy vs. running time: Fig. 6 plots the accuracy (AUC) of each method vs. its running time (in seconds). We see that SL achieves the highest accuracy (=0.91), 8.4% higher than EW (=0.83) and 30% higher than RHSS (=0.70). This gain comes at a cost of a mere 4× slow down compared to EW and RHSS. STA, which computes graph decomposition, was considerably slower.

Accuracy w.r.t. sampling probabilities p, q: Fig. 7a shows how the accuracy varies with source (p) and destination (q) sampling probabilities for K=10 dimensions, after tying p=q for simplicity. We see that the poor accuracy results from choosing very low (anomalous dense subgraphs are easily missed as very few nodes are sampled resulting in a sketch with mostly zeroes) and very high (sketch dimensions are coarse-grained, similar to EW, as almost all nodes are sampled) node sampling probabilities. The sweet spot lies in between. Over a large interval $[0.05, 0.4]$, the accuracy remained fairly robust (insensitive) to the exact value of p.

Accuracy w.r.t. #dimensions K: Fig. 7b shows the variation of accuracy with the number of sketch dimensions $K \in \{5, 20, 35, 50, 65, 80, 95\}$ for $p = q = 0.1$. We see that accuracy increases rapidly from 0.67 to 0.95 as K is increased from 5 to 50, beyond which it stabilizes around 0.95. This is the classic 'diminishing returns' pattern we expect. When K is low, an added SpotLight sketch dimension likely 'illuminates' a new part of the graph and detects anomalies that were previously undetected, but once K crosses a threshold (here, 50) when most of the graph is already 'illuminated', a new sketch dimension gives little to no added benefit.

Q2. Scalability

Fig. 8 shows the scalability of SL with the number of edges and sketch dimensions. We use RRCF with 10 trees and sample size 128.

With #edges: We uniformly sample $100K - 2M$ edges from the Darpa dataset in eight logarithmic steps and timed SL. Fig. 8a plots

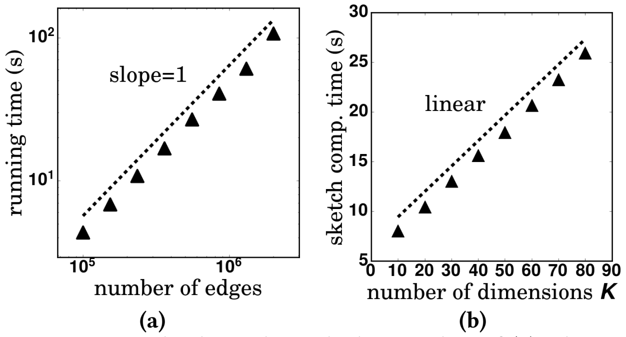


Figure 8: SL scales linearly with the number of (a) edges in the stream and (b) sketch dimensions K.

the running time (in seconds) vs. the number of edges in log-log scales. We see that the points align with a line of slope 1, indicating SL scales linearly with input size (as is desirable). Note also that SL is fast and is able to process $2M$ edges in less than 2 minutes!

With #dimensions: We now vary the SpotLight sketch dimension $K \in \{10, 20, \ldots 70, 80\}$ and measure the time taken to compute sketches for $0.5M$ edges. Fig. 8b, plotting the running time (in seconds) with the number of dimensions, reveals that SL scales linearly with the dimensionality of SpotLight sketch.

These are consistent with our expectations based on Lem. 5.5.

Q3. Discoveries

We provide a complete analysis of SL and baselines on the labeled Darpa dataset; in the interest of space, we only summarize the discoveries due to SL on Enron and NycTaxi datasets, omitting baseline results.

6.2.1 Darpa. Leveraging ground truth, we now delve deeper into why the baselines perform poorly compared to SL on Darpa dataset. Fig. 9 plots the anomaly scores (higher is anomalous) of all methods along with ground truth (spikes in the 'ideal' black curve). Our explanation will use Fig. 10, which plots the number of attack (red) and non-attack (green) edges over time t. In these figures, $t < 0$ corresponds to the initialization period for SL and EW, resulting in zero anomaly score. We now examine each baseline separately.

EW: Around t={150, 450, 650, 850, 1000}, Fig. 10 shows several spikes (of height 10^4–10^5) in attack weight (red); these are significantly higher than the non-attack weight (green) which never exceeds 10^4. Hence, these 'large' anomalies are easily detected by tracking only the total edge weight (green spikes in Fig. 9). However, EW fails to detect anomalous graphs in which the total weight of edges is comparable to that observed at many prior graphs – e.g., anomalies around t={1, 300, 500}. On the other hand, SL keeps track of the total weight of edges *in several local regions within the graph*; since attack edges are concentrated in regions of the graph where non-attack edges typically do not occur, these are detected by SL, even if the weight of attack edges is small, e.g., at t=1.

RHSS: RHSS scores each graph based on the likelihood of its edges computed based on its prior occurrence, preferential attachment and homophily. Simply put, (graphs containing) edges which are seen before or which connect high degree nodes or nodes having many common neighbors are non-anomalous. However, we find that these assumptions are more suited to slowly-evolving social

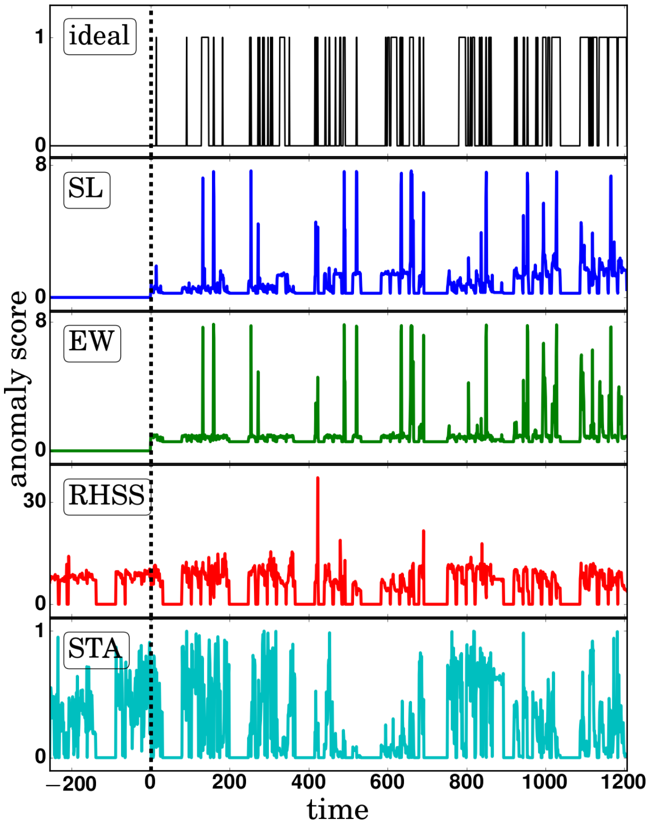


Figure 9: Anomaly detection on Darpa dataset. Spikes in the 'ideal' black curve indicate ground truth anomalies.

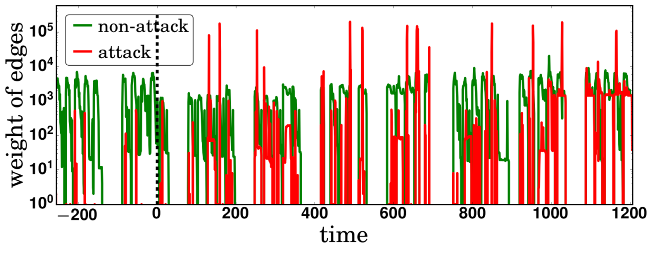


Figure 10: Understanding (un)detected anomalies in Darpa using the number of attack and non-attack edges over time.

networks rather than highly dynamic settings. To see why, consider: (a) *Repeated attacks:* `neptune`[3] attack occurs at 33 different times, including $t = -204$, which is within the initialization period. Once RHSS has 'seen' all `neptune` attack edges, subsequent occurrences, however rare and dense, are not found anomalous. (b) *Repeatedly attacking (victimized) nodes:* Once a node has (been) attacked sufficiently many times, it attains a high degree; consequently, further attacks by (or on) it are 'likely' (due to preferential attachment) and non-anomalous.

STA: STA computes *a single* graph decomposition model to summarize the data seen so far – admittedly, a much harder problem than anomaly detection – and scores the anomalousness of each

[3]A SYN flood denial of service attack to which every TCP/IP implementation is vulnerable to some extent. See www.ll.mit.edu/ideval/docs/attackDB.html.

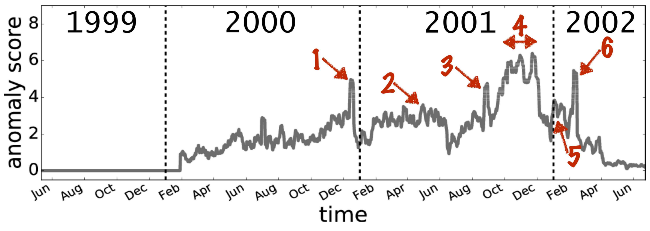


Figure 11: Anomaly detection on Enron dataset

graph as the error incurred in reconstructing it from the model. The assumption of a single normal behavior does not apply to dynamic settings (such as this) – e.g., in Fig. 10, it is as normal for the number of non-attack edges to be around 1000 as it is to be 0 – consequently, STA is very sensitive in practice and leads to numerous false alarms.

6.2.2 Enron. Fig. 11 plots the anomaly score vs. time for Enron dataset, after initializing SL based on the first 256 days (05/12/99-01/22/00) with shingle length 7 (weekly periodicity). We examine the top 6 non-consecutive time durations having the highest anomaly scores. As we show below, these anomalies correspond to major events – either company-wide emails or public announcements triggering excitement or confusion – in the Enron time line[4].

2000: (1) Dec 13-14: Skilling announced as CEO. **2001:** (2) May 23: Enron completes its millionth transaction via Enron Online. (3) Sep 28: Lay to employees: 'Third quarter is looking great.' (4) Oct 7-Nov 22: Wall Street Journal article reveals Enron's precarious state. One ton Enron documents shredded. Fastow ousted. SEC launches formal investigation. Restructuring of $690M obligation is announced. **2002:** (5) Jan 23-30: Lay resigns as chairman and CEO. Baxter commits suicide. Cooper takes over as CEO. (6) Feb 7-8: Fastow, Kopper and Skilling testify before Congress.

6.2.3 NycTaxi. Fig. 12 plots the anomaly score vs. time for NycTaxi dataset, after initializing SL based on the first 256 hours (∼ 10 days) of Nov 2015 with shingle length 24 (daily periodicity). As before, we examine the top 6 non-consecutive time durations having the highest anomaly scores.

The most anomalous period (Jan 23-24) coincided with the January 2016 United States blizzard which produced a historic 3 feet of snow and rendered normal traffic operation impossible. The next three anomalies (around Nov 27, Dec 25, Jan 1) corresponded to festival periods – Thanksgiving, Christmas, New Year – presumably due to unusual traffic patterns around Manhattan (closed offices, Macy's Thanksgiving parade, New Year parties) and airports (people flying in/out of JFK and LaGuardia). The next two anomalies (Nov 14, Nov 29-30) are more interesting because they do not coincide with holidays or weather conditions, and as such, are not expected to be anomalous.

To further understand why Nov 14 and Nov 29-30 were flagged, we derive an anomaly score per sketch dimension from RRCF and *propagate the anomalousness* to NYC zones. Thus, the anomaly score of a zone is the sum of anomaly scores of all dimensions it participates in. The most anomalous zones during these dates turned out to be Bedford on Nov 14 and LaGuardia airport on Nov 29-30. Digging deeper, we discovered that these locations popped up in several archived new articles on these dates. At 12pm Nov 14, 'huge

[4]verified using www.agsm.edu.au/bobm/teaching/BE/Enron/timeline.html

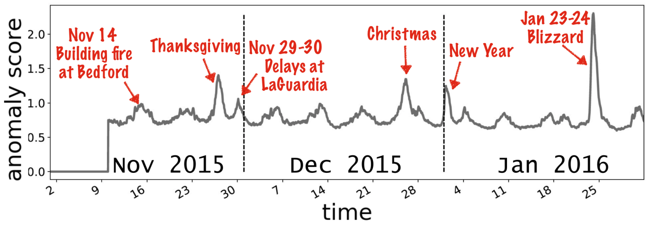


Figure 12: Anomaly detection on NycTaxi dataset

fire [ripped] through Bedford-Stuyvesant building'[5] threatening its collapse and creating unusual traffic in/out of the area. On Nov 29-30 (Sunday after Thanksgiving), 'thousands [were] delayed at airport in an attempt to return home after Thanksgiving'[6] causing the usual morning rush hour traffic at LaGuardia to persist throughout the day with over an hour-long wait times for taxis.

Thus, the *sudden (dis)appearance of large dense subgraphs* detected by SL on real-world data have a practical significance, from network attacks in IP-IP communication logs to holidays, abnormal weather or local traffic conditions in transportation logs.

6.3 Discussion

Why do SL/EW perform better than STA/RHSS? STA and RHSS make strict modeling assumptions, e.g., stable community structure or homophily, restricting their scope to limited settings, e.g., slowly evolving graphs, friendship networks. In contrast, EW and SL use a less restrictive definition of anomaly which is applicable to a wider variety of highly dynamic settings. **Can the detected anomalies be attributed to few nodes?** Yes, by explicitly maintaining the node to sketch dimension mapping and following the 'anomalousness propagation' heuristic in Sec. 6.2.3.

7 CONCLUSION AND FUTURE WORK

We presented a simple, scalable, easy-to-code algorithm called SpotLight for sketching a graph. SpotLight sketches facilitate fast and reliable identification of anomalies, where an anomaly is the sudden appearance (or disappearance) of a large dense directed subgraph. Theoretical analysis provides concrete settings where there is a provable distance gap in the sketch of a graph where m edges are scattered at random throughout the graph (dispersed) vs. the sketch of a graph where m edges are added in a smaller subgraph (focused). The distance gap sets the stage for classic anomaly detection algorithms to spot the more distant graph. Experiments on a variety of real-world datasets demonstrate that SpotLight outperforms prior approaches in terms of both precision and recall. Yet, many new opportunities remain. Adaptive data-driven sketches, while harder to analyze, may yield better results in practice. Interpretability and anomaly attribution are also important questions. Finally, the trajectory of an anomaly is vital to both understand and predict.

ACKNOWLEDGMENTS

We thank Roger Barga, Charles Elkan, Nima Sharifi Mehr, Morteza Monemizadeh and Yonatan Naamad for insightful discussions.

[5]www.nydailynews.com/new-york/huge-fire-rips-bedford-stuyvesant-building-article-1.2435059

[6]pix11.com/2015/11/29/thousands-delayed-at-airport-in-an-attempt-to-return-home-after-thanksgiving/

REFERENCES

[1] 2018. NYC Taxi & Limousine Corporation - Trip Record Data. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.
[2] Charu C. Aggarwal, Yuchen Zhao, and Philip S. Yu. 2011. Outlier detection in graph streams. In *ICDE*. IEEE, 399–409.
[3] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2010. oddball: Spotting Anomalies in Weighted Graphs. In *PAKDD*, Vol. 6119. Springer, 410–421.
[4] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data Min. Knowl. Discov.* 29, 3 (2015), 626–688.
[5] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. CopyCatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*. ACM, 119–130.
[6] Reinhard Diestel. 2012. *Graph Theory, 4th Edition*. Graduate texts in mathematics, Vol. 173. Springer.
[7] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. 2017. ZooBP: Belief Propagation for Heterogeneous Networks. *PVLDB* 10, 5 (2017), 625–636.
[8] Timothy La Fond, Jennifer Neville, and Brian Gallagher. 2014. Anomaly Detection in Dynamic Networks of Varying Size. *CoRR* abs/1411.3749 (2014).
[9] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. 2016. Robust Random Cut Forest Based Anomaly Detection on Streams. In *ICML*, Vol. 48. JMLR.org, 2712–2721.
[10] Keith Henderson, Tina Eliassi-Rad, Christos Faloutsos, Leman Akoglu, Lei Li, Koji Maruhashi, B. Aditya Prakash, and Hanghang Tong. 2010. Metric forensics: a multi-level approach for mining volatile graphs. In *KDD*. ACM, 163–172.
[11] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. FRAUDAR: Bounding Graph Fraud in the Face of Camouflage. In *KDD*. ACM, 895–904.
[12] Tsuyoshi Idé and Hisashi Kashima. 2004. Eigenspace-based anomaly detection in computer systems. In *KDD*. ACM, 440–449.
[13] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2015. A General Suspiciousness Metric for Dense Blocks in Multimodal Data. In *ICDM*. IEEE, 781–786.
[14] Danai Koutra, Neil Shah, Joshua T. Vogelstein, Brian Gallagher, and Christos Faloutsos. 2016. DeltaCon: Principled Massive-Graph Similarity Function with Attribution. *TKDD* 10, 3, 28:1–28:43.
[15] Richard Lippmann, Robert K. Cunningham, David J. Fried, Isaac Graf, Kris R. Kendall, Seth E. Webster, and Marc A. Zissman. 1999. Results of the DARPA 1998 Offline Intrusion Detection Evaluation. In *Recent Advances in Intrusion Detection*.
[16] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *ICDM*. IEEE, 413–422.
[17] Emaad A. Manzoor, Sadegh M. Milajerdi, and Leman Akoglu. 2016. Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs. In *KDD*. ACM, 1035–1044.
[18] Andrew McGregor. 2014. Graph stream algorithms: a survey. *SIGMOD Record* 43, 1 (2014), 9–20.
[19] Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. 2015. Densest Subgraph in Dynamic Graph Streams. In *MFCS*, Vol. 9235. Springer, 472–482.
[20] Tomás Pevný. 2016. Loda: Lightweight on-line detector of anomalies. *Machine Learning* 102, 2 (2016), 275–304.
[21] B. Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. 2010. EigenSpokes: Surprising Patterns and Scalable Community Chipping in Large Graphs. In *PAKDD*, Vol. 6119. Springer, 435–448.
[22] Stephen Ranshous, Steve Harenberg, Kshitij Sharma, and Nagiza F Samatova. 2016. A Scalable Approach for Outlier Detection in Edge Streams Using Sketch-based Approximations. In *SDM*. SIAM, 189–197.
[23] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. 2015. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics* 7, 3 (2015), 223–247.
[24] Jitesh Shetty and Jafar Adibi. 2004. The Enron email dataset database schema and brief statistical report. *Information sciences institute technical report, University of Southern California* 4, 1 (2004), 120–128.
[25] Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2016. M-Zoom: Fast Dense-Block Detection in Tensors with Quality Guarantees. In *ECML/PKDD*, Vol. 9851. Springer, 264–280.
[26] Kumar Sricharan and Kamalika Das. 2014. Localizing anomalous changes in time-evolving graphs. In *SIGMOD*. ACM, 1347–1358.
[27] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. 2007. GraphScope: parameter-free mining of large time-evolving graphs. In *KDD*. ACM, 687–696.
[28] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. 2006. Beyond streams and graphs: dynamic tensor analysis. In *KDD*. ACM, 374–383.
[29] Ke Wu, Kun Zhang, Wei Fan, Andrea Edwards, and Philip S. Yu. 2014. RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection. In *ICDM*. IEEE, 600–609.
[30] Yiming Yang and Xin Liu. 1999. A Re-Examination of Text Categorization Methods. In *SIGIR*. ACM, 42–49.
[31] Weiren Yu, Charu C Aggarwal, Shuai Ma, and Haixun Wang. 2013. On anomalous hotspot discovery in graph streams. In *ICDM*. IEEE, 1271–1276.

APPENDIX (PROOFS FROM SEC. 5.1)

Let $1 \le k \le K$ be a sketch dimension with query subgraph $(\mathcal{S}'_k, \mathcal{D}'_k)$. Define binary random variables $r_{ks} = \mathbb{I}[s \in \mathcal{S}'_k]$ and $u_{kd} = \mathbb{I}[d \in \mathcal{D}'_k]$, where $\mathbb{I}[\cdot]$ is the identity function.

Proof of Lem. 5.1. From Fig. 4a, $v_k(\mathcal{G}_S) - v_k(\mathcal{G}) = \sum_{i=1}^{m} r_{ks_1} u_{kd_i}$. Thus, $\bar{d}(\mathcal{G}_S, \mathcal{G}) = \sum_{k=1}^{K} \mathbb{E}\left[r_{ks_1}(\sum_{i=1}^{m} u_{kd_i})^2\right] = Kmpq + Km(m-1)pq^2$. From Fig. 4b, $v_k(\mathcal{G}_M) - v_k(\mathcal{G}) = \sum_{i=1}^{n} r_{ks_i} u_{kd_i}$ and so we have $\bar{d}(\mathcal{G}_M, \mathcal{G}) = \sum_{k=1}^{K} \mathbb{E}\left[(\sum_{i=1}^{m} r_{ks_i} u_{kd_i})^2\right] = Kmpq + Km(m-1)p^2q^2$. Thus, $\bar{d}(\mathcal{G}_S, \mathcal{G}) > \bar{d}(\mathcal{G}_M, \mathcal{G}) + O\left(Km^2\right)$. ■

The other results are based on Lem. .1 stated and proved below.

Lemma .1. *Let $\mathcal{G}$ be an arbitrary graph and let $\mathcal{G}'$ be obtained by adding $m(\le n^2)$ unit-weight edges (in expectation) uniformly to any $n \times n$ region of $\mathcal{G}$ by sampling each of the n^2 possible edges independently with probability m/n^2. Assuming n is large and p=q,*

$$\mathbb{E}\left[\bar{d}(\mathcal{G}, \mathcal{G}')\right] = Kp^2m\left(1 + \frac{2pm}{n} + p^2m\right) \tag{4}$$

Further, if $n \gg m$, Var $\left[\bar{d}(\mathcal{G}, \mathcal{G}')\right] = O\left(Kp^4m^2\left(1 + 2p^2m + p^4m^2\right)\right)$, where the expectation and variance have been taken over the random coin tosses of the edge addition process.

Partial Proof. Let $\mathbf{A} = [A_{sd}]$ denote the adjacency of edges added to $\mathcal{G}$ to get $\mathcal{G}'$. Then, $\bar{d}(\mathcal{G}, \mathcal{G}') = \mathbb{E}\left[(\sum_{s,d} r_{ks} A_{sd} u_{kd})^2\right]$, where the expectation is taken over the coin tosses of the algorithm, i.e., $\{r_{ks}, u_{kd}\}\ \forall\, k, s, d$. Using $\mathbb{E}[r_{ks}] = \mathbb{E}[u_{kd}] = p$. This simplifies to $\bar{d}(\mathcal{G}, \mathcal{G}') = p^2 \sum_{s,d} A_{sd} + p^3 \sum_{s,d \ne d'} A_{sd} A_{sd'} + p^3 \sum_{s \ne s',d} A_{sd} A_{s'd} + p^4 \cdot \sum_{s \ne s', d \ne d'} A_{sd} A_{s'd'}$. To get $\mathbb{E}\left[\bar{d}(\mathcal{G}, \mathcal{G}')\right]$ where the expectation is now taken over the randomness of edge addition, i.e., A_{sd}, we substitute $\mathbb{E}[A_{sd}] = m/n^2$ for $1 \le s \le n, 1 \le d \le n$ (and otherwise zero) to derive Eq. (4). Variance calculation, while similar and straight-forward, is omitted in the interest of space. ■

Proof of Thm. 5.2 using Lem. .1. Eq. (4) is decreasing in n. ■

Proof of Thm. 5.3 using Lem. .1. Let $\mu = \mathbb{E}\left[\bar{d}(\mathcal{G}, \mathcal{G}_{\text{ER}})\right]$ and $\sigma^2 = \text{Var}\left[\bar{d}(\mathcal{G}, \mathcal{G}_{\text{ER}})\right]$. Invoking Chebyshev's inequality, we have with probability $1-\delta$: $|\bar{d}(\mathcal{G}, \mathcal{G}_{\text{ER}}) - \mu| \le \sqrt{\sigma^2/\delta}$. Thus, if we flag a graph $\mathcal{G}'$ as anomalous if $|\bar{d}(\mathcal{G}, \mathcal{G}') - \mu| > \sqrt{\sigma^2/\delta}$, we erroneously flag δ fraction of the control group as anomalies (false positive rate). In order to detect $\mathcal{G}_{\text{BC}}$ as an anomaly at this threshold, we need $\bar{d}(\mathcal{G}, \mathcal{G}_{\text{BC}}) - \mu - \epsilon > \sqrt{\sigma^2/\delta}$. Under the stated assumptions, $\bar{d}(\mathcal{G}, \mathcal{G}_{\text{BC}}) - \mu \approx 2Kp^3n^3$ and $\sigma^2 \approx Kp^4n^4(1 + 2n^2p^2 + n^4p^4)$. Thus, we derive a quadratic inequality in K resulting in the following, which can then be relaxed using $a^2 + b^2 \ge 2ab$ to obtain Eq. (3). ■

$$K \ge \left(\frac{1 + n^2p^2}{4pn\sqrt{\delta}} + \sqrt{\left(\frac{1 + n^2p^2}{4pn\sqrt{\delta}}\right)^2 + \frac{\epsilon}{2p^3n^3}}\right)^2$$

Proof of Cor. 5.4. Setting the first derivative of RHS of Eq. (3) to zero, we get $n^5p_*^5 - np_* = 6\epsilon\delta$ (second derivative at $p_* \ge 0$). ■