A Scalable Approach for Outlier Detection in Edge Streams Using Sketch-based Approximations

Stephen Ranshous* Steve Harenberg* Kshitij Sharma* Nagiza F. Samatova*†‡

Abstract

Dynamic graphs are a powerful way to model an evolving set of objects and their ongoing interactions. A broad spectrum of systems, such as information, communication, and social, are naturally represented by dynamic graphs. Outlier (or anomaly) detection in dynamic graphs can provide unique insights into the relationships of objects and identify novel or emerging relationships. To date, outlier detection in dynamic graphs has been studied in the context of *graph streams*, focusing on the analysis and comparison of entire graph objects. However, the volume and velocity of data are necessitating a transition from outlier detection in the context of graph streams to outlier detection in the context of *edge streams*–where the stream consists of individual graph edges instead of entire graph objects.

In this paper, we propose the first approach for outlier detection in edge streams. We first describe a high-level model for outlier detection based on global and local structural properties of a stream. We propose a novel application of the Count-Min sketch for approximating these properties, and prove probabilistic error bounds on our edge outlier scoring functions. Our sketch-based implementation provides a scalable solution, having constant time updates and constant space requirements. Experiments on synthetic and real world datasets demonstrate our method's scalability, effectiveness for discovering outliers, and the effects of approximation.

1 Introduction

Dynamic graphs, which allow modifications to the structure and/or attributes of the graph over time, have become a very popular way to model an evolving set of objects and their ongoing interactions. A common example is Facebook, where vertices in the graph are the users and the edges can represent friendship, wall posts, chat messages, or a number of other phenomena. As a result of how ubiquitous dynamic graphs have become, there has been a surge of research in developing techniques to attain novel insights into the represented system. In particular, outlier (or anomaly) detection has become a topic of great interest given its many applications [18]. For example:

- Information networks, such as IMDB, DBLP, or the US patent citation networks, can provide insight into new collaborations. Edges that bridge two actors from conventionally different genres of movies in IMDB (or two authors from different disciplines of computer science in DBLP, or engineers from different fields in the citation network) indicate a deviation from the standard patterns and show interesting, new, cross-discipline projects.
- Social networks, such as Twitter or Facebook, can connect users based on various interaction types such as friendships, mentions, or wall posts. Outliers can constitute shifts in friendship, or new trending persons or topics.

Typically, these dynamic graphs are broken into a sequence or stream, of static graphs, and analysis is performed by comparing adjacent graphs in the stream[10, 17, 19]. However, there are several limitations to performing outlier detection in this manner. First, it requires that entire graphs be stored either in main memory or offline for analysis, which is infeasible given the large universe from which edges and vertices are drawn from and the rate at which the graphs change. For example, networks such as Twitter or Facebook can have on the order of 10^6–10^9 active users (meaning 10^{12}–10^{18} possible unique edges), with Twitter processing 500 million tweets per day in 2013[1] and 293,000 status updates every 60 seconds on Facebook[2]. Second, much or all of the historical information of the stream is lost, as only the most recent graphs in the stream are considered. Third, global comparisons of the graphs often precludes the possibility of performing attribution for the changes in the graph. In response to these limitations, we propose a transition to outlier detection in the context of *edge streams*, where objects in the stream are individual edges from the graph instead of full graph objects.

As edge stream mining is a more fine-grained analysis of the changes, we hypothesize it is more much amenable to attribution. However, addressing memory constraints and historical information loss issues is still a challenging problem. Two common approaches for addressing these issues in stream mining are sampling and sketches [2]. Sampling attempts to retain the salient

*North Carolina State University, Raleigh, NC
†Oak Ridge National Laboratory, Oak Ridge, TN
‡Corresponding author: samatova@csc.ncsu.edu

[1] http://www.internetlivestats.com/twitter-statistics/
[2] https://zephoria.com/social-media/top-15-valuable-facebook-statistics/

portions of data from the stream for analysis, while forgetting (not storing) the rest. The main challenge is how to determine what information is important in a single pass, and how to remain agnostic to the order in which the data is seen. Conversely, sketches maintain a compressed summary of the *entire* data stream. In this work, we propose a model for calculating an outlier score for each edge in a continuous edge stream, and we utilize the Count-Min (CM) sketch [9] to approximate the structural properties of the stream that are relevant to this model. Using this approximation, we are able to prove probabilistic error bounds on our edge scores.

We propose, to the best of our knowledge, the first method for identifying outliers in an edge stream. Our major contributions can be summarized as follows.

1. **Scalable Streaming Model for Outlier Detection.** We describe a data driven model for scoring edges in a continuous stream (Section 2). We propose a sketch-based approach that uses a constant amount of space that is independent of the size of both the graph and the stream itself, and performs updates and outlier analysis in constant time (Section 3, Appendix B).
2. **Theoretical Bounds.** We show theoretical probabilistic bounds for the error of the estimations of our sketch-based approach (Section 3).
3. **Extensive Experiments.** Extensive experiments are performed to test scalability, measure the true error resulting from approximation, empirically test the effects of approximation on identifying injected ground-truth outliers, and validate the models ability to detect outliers in real-world data (Section 4).

2 Problem Statement and Model

We informally define the problem we address as follows.

PROBLEM 1. (OUTLIER DETECTION IN EDGE STREAMS)
Given: A continuous edge stream $\mathcal{E} =< (u, v), ... >$, where u and v are the vertices the edge is incident upon.
Find: Edges in the stream that deviate from the expected structural patterns.

It is important to note the distinction between this formulation and the problem of link prediction. In link prediction, outliers are when edges that *are expected* to occur *do not*, while we define outliers as edges that *are not expected* to occur *but do*. For example, edges that occur in the stream that bridge two previously disconnected components of the graph would be an outlier by our problem definition, but not in the link prediction problem (see Section 1 for real-world examples). To this end, we propose a data-driven structural model that encapsulates both global edge characteristics and vertex-level egocentric behaviors.

2.1 Data Model Before discussing the structural model, it is essential to understand the type of data we will be modeling. Every edge in the stream is undirected[3], unweighted/unlabeled, and represents the insertion of a single edge into the graph. This is a special case of the "cash register model" [16] for streams. $\mathcal{E}$ represents a multigraph, therefore the number of times an edge may occur in the stream is unrestricted. Similarly, no *a priori* knowledge on the total number of vertices in the graph is required, and the total vertex set may (and likely will) grow over time. Every vertex v is assumed to be represented by a unique label (e.g., user names or IP addresses), $l(v)$, thus $l(u) = l(v)$ iff $u = v$. Edges are labeled as the concatenation of the two vertices they are incident upon, thus the label of $e = (u, v)$ is $l(e) = l(u) \oplus l(v)$, where $\oplus$ is the concatenation operator.

2.2 Edge Scoring and Outlier Detection Given this data model, the goal is to identify edges that appear in the stream that deviate from the historical patterns of the graph, and thus the expected behavior of the incoming stream. Every incoming edge is scored based on historical evidence as well as connectivity patterns of the vertices it is incident upon. The score of an edge scales proportionally with the extent to which it conforms to the expected behavior, low scores representing outliers and high scores representing "normal" edges. This scoring, and consequently the labeling of an edge as an outlier or not, must be done in real-time. Moreover, given the vast size of the networks and the rate at which updates occur, the analysis should only require space that is sublinear, or independent of, the size of the network and stream. To achieve this, we propose a sketch-based approach for probabilistically modeling the edge stream with provable error bounds (see Section 3). However, for simplicity and clarity when describing the high level outlier detection model, in this section, we assume that the stream is being stored exactly and explicitly using an adjacency matrix A.

The structural properties of the edge stream will be modeled using a weighted graph, where every incoming edge $(u, v) \in \mathcal{E}$ increments the corresponding edge weight A_{uv} by 1. Thus, if (u, v) has occurred in $\mathcal{E}$ three times, $A_{uv} = 3$. As we consider the undirected case, A is symmetric, $A_{uv} = A_{vu}$. We define the weight of a vertex as the sum of all of its edge weights, $w(u) = \sum_v A_{uv}$, and, abusing notation, we define the weight of an edge as $w(u, v) = A_{uv}$. For a given vertex u, its neighborhood is defined as the set of all vertices which are adjacent to it, $N(u) = \{v \mid (u, v) \in \mathcal{E}\} = \{v \mid A_{uv} > 0\}$.

[3]Our model is easily generalized to directed edges.

Evidence-Based Scoring. One of the strongest indicators of an edge occurring in the stream is its historical presence. As discussed in [5], two dominating properties of dynamic real-world networks are their power law degree distributions and the self-similarity of edge additions. Intuitively, this equates to edges that have occurred more often having a higher probability of occurring again. This intuition leads us to a simple sample-based scoring function.

DEFINITION 2.1. (SAMPLE SCORE) *The sample score of an edge (u, v), s^s_{uv}, occurring in the edge stream $\mathcal{E}$ is proportional to the weight of the existing edge between u and v compared to the total weight of the graph.*

$$s^s_{uv} = \frac{A_{uv}}{\sum_i \sum_{j>i} A_{ij}} = \frac{A_{uv}}{|\mathcal{E}|} \tag{2.1}$$

However, sample-based scoring does not account for the sparsity of real-world networks, where most vertex pairs are not connected. To smooth the sample-based scoring, we introduce a second scoring function. We use a normalized version of the preferential attachment score, commonly used in the link prediction community [14, 15] and made popular by the Barabási-Albert graph generation model [7]. This non-sample-based score scales proportionally with the degree of the two vertices it is incident upon, meaning edges that have never been seen before can be assigned a non-zero score. Normalizing it ensures that every score is mapped into the range [0, 1], like the sample score is.

DEFINITION 2.2. (PREFERENTIAL ATTACHMENT SCORE) *The preferential attachment score of an edge (u, v), s^p_{uv}, occurring in the edge stream $\mathcal{E}$ is proportional to the product of the weights of u and v. Let $||A_{u\cdot}|| = \sum_v A_{uv}$.*

$$s^p_{uv} = \frac{||A_{u\cdot}|| \cdot ||A_{v\cdot}||}{|\mathcal{E}|^2} \tag{2.2}$$

Note that this is a monotonically decreasing function for each edge. As more edges are seen in the stream, and the graph model grows more robust, the data-driven sample score begins to dominate over this smoothing score.

Neighborhood-Based Scoring. Evidence-based scoring functions provide accurate data-driven metrics for assessing the degree to which an edge fits the global expectation. Nevertheless, they fail to capture the edges fit when considering the local surrounding structure. Equivalently, they capture only *direct* connections, and fail to consider the *indirect* connections of vertices. For example, consider vertex 2 in Figure 1. Vertices 0 and 2 are connected by a heavily weighted edge, indicating that $(0, 2)$ has been seen in the stream numerous times. Therefore, it is expected that there will be more

Edge	s^s	s^p	s^h	s
(2, 0)	.300	.060	.000	.270
(2, 1)	.000	.060	1.00	.295
(2, 3)	.100	.030	.000	.098
(2, 4)	.100	.030	.000	.098
(2, 5)	.100	.030	.000	.098

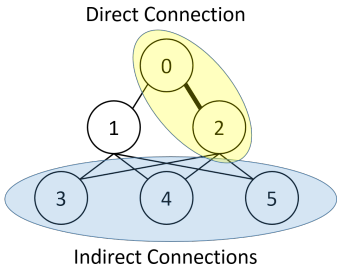

Figure 1: Example graph and table illustrating the scores each edge incident to vertex 2 would receive if it were to arrive next in the stream. For calculations, $w(0, 2) = 3$, all other edge weights are 1, $\alpha + \beta = 0.75$, and $\gamma = 0.25$.

edges between them in the future. Conversely, vertices 1 and 2 have no edge between them, indicating that $(1, 2)$ has not been in the stream. Would it be strange then for an edge connecting them to appear? Using only sample-based scoring – yes. However, given the number of shared neighbors, it should not be surprising. The expectation that two vertices with many common neighbors will interact is a manifestation of the phenomenon known as homophily. In general, homophily is the tendency for objects to interact with other objects that are similar to themselves.

We derive a metric to quantify the similarity between two vertices by extending a simple metric commonly used in the link prediction literature. Given two vertices, the Jaccard Similarity Coefficient of their neighborhoods is a simple yet effective way to express the probability of having an edge between them. However, this metric treats all neighbors equally, ignoring the weights of their connections. Therefore, we extend this metric by proposing a *weighted* neighborhood intersection and an appropriate normalization.

DEFINITION 2.3. (HOMOPHILY SCORE) *The homophily score of an edge (u, v), s^h_{uv}, occurring in the edge stream $\mathcal{E}$ is proportional to the weight of the intersection of the neighborhoods of u and v.*

$$s^h_{uv} = \frac{\sum_{k \in N(u) \cap N(v)} (A_{uk} + A_{vk})}{||A_{u\cdot}|| + ||A_{v\cdot}||} \tag{2.3}$$

Using Eqn. 2.3, not only is the size of the intersection considered, but the proportional weight as well. Moreover, the direct connection between the vertices incident on the edge is not included, removing the potential overlap with the sample score. As desired, given two vertex pairs with equal neighborhood intersection sizes, the pair with a higher percentage of their summed vertex weights on edges incident on vertices in the intersection will have a higher homophily score.

Total Score. Every edge is mapped to a single score, indicating how well it is in accordance with the expected structural behavior. We define the total score for an edge as the linear combination of the three above-mentioned scoring functions.

DEFINITION 2.4. (TOTAL EDGE SCORE) *The total score of an edge (u, v), s_{uv}, occurring in the edge stream $\mathcal{E}$ is a linear combination of the evidence- and neighborhood-based scoring functions.*

$$s_{uv} = \alpha s^s_{uv} + \beta s^p_{uv} + \gamma s^h_{uv} \text{ where } \alpha + \beta + \gamma = 1.$$

The coefficients α, β, and γ serve as weights between the scores, giving applications the flexibility to value, for example, the sample score over the homophily score. As all three scoring functions map to a real number in [0,1], the total score is also in the range [0,1]. See Figure 1 for an example graph with all the scores shown.

Outlier Detection. By mapping every edge to a real valued number, we have shifted the domain for outlier detection from graphs to time series. Outlier detection in time series data is a well-studied domain [13]. In this work we choose a simple thresholding approach, but any approach that can evaluate a new data point in constant time is suitable. We can now restate Problem 1 more formally. Given a continuous edge stream $\mathcal{E}$, and a threshold score τ, identify all outlier edges $(u, v) \in \mathcal{E}$ such that $s_{uv} < \tau$.

3 Model Approximation

Explicit and exact storage of the network structure for on or offline analysis is infeasible for streams at even a moderate scale. In [4] the authors introduce a structural reservoir sampling technique to heuristically store sub-structures in the graph to facilitate analysis on multiple samples of the graph. We propose a complementary method that uses a probabilistic data structure, known as the Count Min (CM) sketch, to store an approximation of the complete stream. Sketch-based techniques are commonly used in streaming and time series scenarios to maintain important statistics of the overall stream in a greatly reduced space [2]. We will focus on the Count Min sketch [9], and how it can be used to store the required edge frequencies, weights, and vertex neighborhood information for our model. In this section, we describe how CM sketches are used, and we prove probabilistic guarantees for the accuracy of the scoring functions when using the sketch-based model.

The Count-Min Sketch. A CM sketch is a two-dimensional array of width w and depth d that is used for storing approximations of data stream frequencies. Frequency queries on CM sketches have known accuracy guarantees that are characterized by the parameters ε and δ, which dictate the error of the approximation and the probability of the approximation being within that error, respectively. More concretely, if $\boldsymbol{a}$ is a vector with the true counts of the objects seen in the stream, then a CM sketch of width $w = \lceil \frac{e}{\varepsilon} \rceil$ and depth $d = \lceil \log \frac{1}{\delta} \rceil$ will give an object count within a factor of $\varepsilon ||\boldsymbol{a}||$ with probability at least $1 - \delta$ [9].

As objects arrive in the stream, d independent pairwise hash functions[4] are applied to the object's unique identifier, and the counts at the corresponding cells in the sketch are updated. The error of the approximation comes from hash collisions, since the width is much smaller than the total number of unique objects in the stream. It is important to note that every frequency approximation, from each of the d rows, will be at least as large as the true count. As a result, the best estimate for the true count of an object is the minimum of the d approximations.

Sample Score. A single sketch is required for estimating the sample score. Recall that every edge has a unique identifier that is the concatenation of the labels of the two incident vertices, $l(e) = l(u) \oplus l(v)$. Thus, an estimation for the number of times every edge has appeared in the stream can be stored. As the number of edges seen in the stream is known exactly, the error is only introduced in the numerator of the scoring function. Using this, we can derive the following probabilistic error bound for the sample score of an edge.

LEMMA 3.1. *Using a Count-Min sketch of width $w = \lceil \frac{e}{\varepsilon} \rceil$ and depth $d = \lceil \ln \frac{1}{\delta} \rceil$, the sample score of an edge (u, v) is bound by the following inequality with probability* at least $1 - \delta$.

$$s^s_{uv} \leq s^{se}_{uv} \leq s^s_{uv} + \varepsilon$$

Proof. This proof, and all subsequent proofs, have been placed in the appendix due to space constraints.

This result implies that a very good estimation is possible using *only a few megabytes* of memory. For example, letting $\varepsilon = .00001$ and $\delta = .001$, only 7.26 megabytes (assuming 32-bit counters) are needed to give a score within .00001 of the true value with probability at least .999.

Preferential Attachment Score. The preferential attachment score is estimated in a very similar manner to the sample score. A single (different) CM sketch will be used to estimate the weight of every vertex in the graph. Again, the denominator is known exactly, resulting in the error being from the numerator only. Unlike the

[4]By pairwise independent, we mean that for a given hash function h_i, $\mathbb{P}[h_i(x) = h_i(y) \mid x \neq y] = \frac{1}{w}$.

sample score, the error is multiplicative since the sketch is used to get the weight of each vertex of the given edge. Multiplicative error withstanding, the following bound is derived for the preferential attachment score.

LEMMA 3.2. *Using a Count-Min sketch of width* $w = \lceil \frac{e}{\varepsilon} \rceil$ *and depth* $d = \lceil \ln \frac{1}{\delta} \rceil$*, the preferential attachment score of an edge* (u, v) *is bound by the following inequality with probability* at least $1 - \delta$*.*

$$s^p_{uv} \leq s^{pe}_{uv} \leq s^p_{uv} + \varepsilon$$

Homophily Score. To estimate the homophily score for an edge, we must somehow maintain approximate neighborhood information for every vertex in the graph. To estimate the neighborhood of a vertex, one option would be to use the sketch for the sample score (i.e., edge weights). Given a vertex u, an approximate neighborhood could then be found by iterating over all edges (u, v); if the weight of an edge is greater than zero, then v is considered part of the neighborhood of u. However, with no information of the existing edges being stored explicitly, all $O(|V|)$ potential neighbors would have to be checked. In a streaming environment, this is unacceptable. Instead, we propose two sketch-based solutions: one with a provable probabilistic accuracy guarantee and one that is a heuristic based on well-known real-world network properties (the effectiveness of each solution is evaluated in Section 4).

Individual Sketches Approximation. Our first approach provides a provable probabilistic error bound. We use a (new) third sketch but, instead of a traditional two-dimensional sketch, we extend the sketch into an additional dimension. Thus, our new sketch is of size $w \times d \times l$, where length l is the new dimension. At any given point in time, l is equal to the number of unique vertices seen in the stream. Equivalently, every time a new vertex is seen, an additional sketch is allocated for it. With each vertex having its own sketch, the width can be substantially smaller than the width of the sketches for the vertex weights and the edge weights. Real-world networks commonly follow a power law degree distribution [7], meaning the majority of nodes have a very small neighborhood. In addition to implying that the sketches can be smaller, it implies that it is rare that two vertices that do not already belong to the same small cluster will share a new edge. This is important because smaller sketches equate to mapping the same potential number of nodes ($|V|$) into a much smaller range, increasing the probability of collisions. The reduced number of edges between distant neighbors results in a decreased amount of error.

LEMMA 3.3. *Assume every vertex has their own sketch to track the weight of the edges between themselves and every vertex in their neighborhood. Given two vertices,* u *and* v*, and their corresponding sketches,* N_u *and* N_v*, each of width* $w = \lceil \frac{e}{\varepsilon} \rceil$ *and depth* $d = \lceil \ln \frac{1}{\delta} \rceil$*, the homophily score of* (u, v) *is bound by the following inequality with probability* at least $1 - \delta$*,*

$$s^h_{uv} \leq s^{he}_{uv} \leq s^h_{uv} + \varepsilon N_{uv}$$

where N_{uv} *is the number of cells that have a nonzero value in both* N_u *and* N_v*.*

Sketch Sharing Heuristic. While providing a provable error bound, allocating a sketch (however small) for each vertex means the required space scales linearly with the number of vertices. However, recalling the sparsity of real-world networks, their degree distributions, and the tendency towards having small cohesive substructures (communities), we can reduce the required space, by allowing vertices to share neighborhood sketches. Again, we will use a (new) third sketch that is extended into three dimensions. However, this time, the length will be a fixed constant that is independent of the number of vertices seen.

Clearly, for any assignment of $|V|$ vertices to $l < |V|$ sketches, at least $\lceil |V|/l \rceil$ vertices will share a sketch. To remove any potential bias and dependency on ordering, we randomly assign the vertices to a sketch by hashing the label of the vertex to a number in [1, l]. Moreover, random assignments minimize the error by keeping the expected number of vertices per sketch at $\lceil |V|/l \rceil$. Even with a fixed length, the expected error can be reduced by modifying the width of the sketch, as the error is a function of the number of hash collisions, and the expected number of collisions is inversely proportional to the width of the sketch.

Total Score. As the total score is a linear combination of the three individual scores, the error is a linear combination of the error of each score. The bound below follows.

THEOREM 3.1. *Using a CM sketch of width* $w = \lceil \frac{e}{\varepsilon_1} \rceil$ *and depth* $d = \lceil \ln \frac{1}{\delta} \rceil$ *for estimating sample scores, a CM sketch of* $w = \lceil \frac{e}{\varepsilon_2} \rceil$ *and* $d = \lceil \ln \frac{1}{\delta} \rceil$ *for estimating preferential attachment scores, and* $|V|$ *CM sketches of* $w = \lceil \frac{e}{\varepsilon_3} \rceil$ *and* $d = \lceil \ln \frac{1}{\delta} \rceil$ *for estimating homophily scores, the total estimated score of an edge* (u, v)*,* s^e_{uv}*, is bound by the following inequality with probability at least* $1 - \delta$*.*

$$s_{uv} \leq s^e_{uv} \leq s_{uv} + \alpha\varepsilon_1 + \beta\varepsilon_2 + \gamma\varepsilon_3 N_{uv} \tag{3.4}$$

where α*,* β*, and* γ *are weighting coefficients such that* $\alpha + \beta + \gamma = 1$ *.*

The bound above holds only when each vertex uses their own neighborhood sketch. When employing the shared

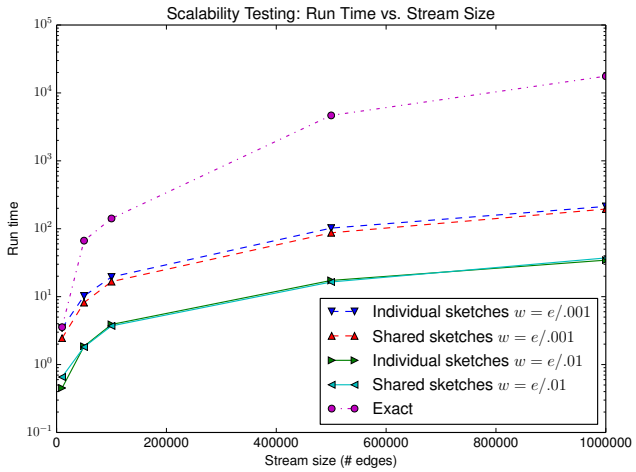

Figure 2: Run times for various approximation parameters compared to the exact implementation. For all approximations, the vertex and edge weight sketches had $\varepsilon = 10^{-6}$ and $\delta = 10^{-3}$. The shared sketch heuristic runs both had a length of $l = \lceil \frac{e}{.0001} \rceil$.

sketch heuristic, the homophily score, and thus the total score, is not bound. However, we show in Section 4 that the heuristic performs very well in practice.

4 Experiments

For our experiments we have implemented the exact, approximate, and heuristic versions (see Appendix C for experimental setup). As a baseline, we implemented the graph stream outlier detection approach proposed in [4]; see Section 5 for more details of this approach. Similar to our edge score, each edge in the baseline is assigned a probability of appearing.

4.1 Scalability To test the scalability of our proposed method, we generated several synthetic datasets using the Random Typing Generator (RTG) [5]. RTG is the state-of-the-art method for generating dynamic networks that follow several important properties present in real-world networks, e.g., power law degree distribution. As such, it enables us to generate realistic datasets for scalability testing.

As shown in Figure 2, the run time increases linearly with the size of the stream, regardless of the chosen parameter values. However, the effect the sketch parameter values have on the size of the constant is evident. As is typical with approximation algorithms, there is a tradeoff between accuracy and efficiency. As the error bounds shrink (ε shrinks) or the probability of the bounds holding increase (δ decreases), the run time increases commensurately. The dominating run time factor for the approximate and heuristic versions is the width of the sketches used for calculating the homophily score. When using equal widths for their neighborhood sketches, the approximate and heuristic versions have similar run times.

In all cases, the exact version took orders of magnitude longer to process the stream. Given the memory constraints of the exact version, it was necessary to store the graph using a sparse matrix representation. As a result, all update operations require significantly more time compared to using a dense matrix, contributing to the slow run time observed. For example, updating the weight of an edge using a dense matrix is $O(1)$, while it is $O(|V|)$ in a sparse matrix (though, typically much better in practice due to network sparsity).

The baseline run time is omitted as our implementation may be less efficient than that of the authors, with ours taking multiple days per edge stream. As the baseline is designed for batch updates, where each new graph in the stream triggers an update, processing edge streams (where each edge is treated as its own graph) is expected to be a worst-case scenario. Each new edge requires multiple edge-set samples to update their spanning forests and the statistics for each connected component.

4.2 Identifying Injected Outliers As no ground-truth data is available for testing the precision of our methods, we must manually create a ground-truth dataset. We follow a method outlined in [6] and inject outliers into the well-studied, real-world Enron email dataset. Artificial outliers are injected by sampling two vertices from a uniform distribution and adding an edge between them. This generation opposes the preferential attachment model, where vertices are connected with probabilities proportional to their degrees and, as such, contradicts the shown power law distribution of the vertex degrees and edge multiplicities [1].

In total, six types of anomalous datasets were created by varying the percent of outlier edges as one or five over all or half of the stream (for full details of all other datasets, see Appendix D). As the injection process was random, 10 edge streams were created for each dataset type, creating a total of 60 anomalous edge streams. All of the results shown for each dataset type are averages taken over the 10 edge lists.

4.2.1 Approximation Effects on Precision We measure the precision of our method by comparing the area under the curve (AUC) of the receiver operating characteristics (ROC) curve for various sketch parameter values. The ROC curve is formed by plotting the true positive rate (TPR) against the false positive rate (FPR) for a range of outlier threshold values. For calculations, we consider the positive class the outlier edges

Table 1: ROC AUC comparisons for vertex and edge weight sketch parameters $\varepsilon = .000001$, with the length of the shared sketch size shown above, all sketches set $\delta = .001$

Injection Type	Exact $w = \text{N/A}$ $l = \text{N/A}$	Baseline $w = \text{N/A}$ $l = \text{N/A}$	Individual sketches $l = \text{N/A}$ $w = \lceil \frac{e}{.001} \rceil$	Individual sketches $l = \text{N/A}$ $w = \lceil \frac{e}{.01} \rceil$	Shared sketches $l = \lceil \frac{e}{.0001} \rceil$ $w = \lceil \frac{e}{.001} \rceil$	Shared sketches $l = \lceil \frac{e}{.0001} \rceil$ $w = \lceil \frac{e}{.01} \rceil$
Full: bursty	0.971951	0.849379	0.974366	0.968807	0.954515	0.923315
Full: 1%	0.972161	0.848936	0.974553	0.969220	0.954966	0.924233
Full: 5%	0.973332	0.864021	0.974064	0.961259	0.952195	0.904643
Half: bursty	0.967197	0.844626	0.968344	0.958802	0.942415	0.899664
Half: 1%	0.967260	0.844871	0.968444	0.959224	0.941580	0.900588
Half: 5%	0.967735	0.844603	0.968337	0.954138	0.939952	0.887038

we manually injected, and the rest of the stream the negative class. The TPR is proportional to the number of edges in the positive class that were correctly classified (labeled) as outliers, and the FPR is proportional to the number of edges in the negative class that were falsely classified as positives (i.e. normal edges being classified as outlier edges). The AUC is thus in the range [0, 1], where 1 is a perfect classifier. Table 1 shows the results for the exact, approximate, heuristic, and baseline implementations.

We first look at the performance of the individual sketch approximation compared to exact. Both achieved very high AUCs, around .97 for all datasets, indicating they were highly successful in identifying the outlier edges. Interestingly, with $w = \lceil \frac{e}{.001} \rceil$, the approximation yielded a higher AUC for all datasets. For this to happen, the error resulting from the sketches must have worked in favor of the classifier. Using a smaller width of $w = \lceil \frac{e}{.01} \rceil$ we do not see this effect, and the AUC is slightly less than the exact version at around .966.

Next, we compare the shared sketch heuristic to the individual sketch approximation. For equal widths, the individual sketches perform better, as expected, due to the smaller error on the homophily scores. A key difference between the two is the effect of the sketch widths. When the individual sketches go from $\varepsilon = 10^{-3}$ to 10^{-2} there is very little change in the AUC. The same change in size causes a drop of around .05 for all datasets in the shared sketch heuristic. Again this is expected, as the number of collisions (i.e. error) due to sharing is proportional to the width of the sketch.

Finally we compare our implementations to the baseline. We modified the baseline method very slightly in order to improve the accuracy: Instead of placing every component smaller than `MIN_COMPONENT_SIZE` into a single outlier component, we left them in their own component. When they were grouped into a single, large outlier component the AUC was near zero, as the outlier edges were connecting vertices in this component, resulting in a very high non-sample score for the edge. With our modification, the AUC for the baseline rose an order of magnitude from .08 to .84. However, both the approximate and heuristic versions of our method outperform the baseline on all datasets. We are able to achieve a higher AUC than the baseline in addition to providing constant space and time complexities (Appendix B). No time or space complexity analysis for the baseline is shown in [4].

4.2.2 Approximation Effects on Scoring

Using a CM sketch to store the edge stream model introduces approximation error to the scores calculated for every edge in the stream. Table 2 shows the error caused by approximation on a variety of datasets.

We focus on the error of the homophily score, as that is the most volatile and parameter sensitive. For all of the Enron datasets, the homophily score has a median value of 0, with a standard deviation of 0 (to the ninth decimal place at least). However, scores for IMDB and DBLP show a very different result; while medians are still 0, standard deviations are on the order of at least 10^{-4}. This large descrepency indicates that there are a few very large hub vertices in the network, causing a high number of collisions in the neighborhood sketches. Most of the vertices have a very accurate picture of their neighborhood, having only a few neighbors, hence the median error being low. Further evidence of this phenomenon is the low error of the individual sketches compared to the shared sketches. No vertex shares a neighborhood sketch with a hub vertex, thus the error is confined to only the edges incident to hub vertices.

4.3 Real-World Outliers

While synthetic datasets are ideal for unit testing components of an algorithm and providing ground truth, it is also illustrative and insightful to examine real-world use cases and outliers. Here we highlight a few examples of real-world outliers found using our approach.

Table 2: Approximation error as a result of using sketches. The values shown for IMDB and DBLP are $median \times 10^6$ ($\pm std \times 10^3$), for the Enron datasets (Full and Half datasets) the values shown are $median \times 10^6$ ($\pm std \times 10^9$). For all runs, the vertex and edge weight sketches had $\varepsilon = 10^{-6}$ and $\delta = 10^{-3}$.

Dataset	Shared sketches: $w = \lceil \frac{e}{.001} \rceil, l = \lceil \frac{e}{.0001} \rceil$				Individual Sketches: $w = \lceil \frac{e}{.001} \rceil, l = \lceil \frac{e}{.0001} \rceil$			
	s^s	s^n	s^h	s	s^s	s^n	s^h	s
IMDB	0.07 (±0.53)	0.07 (±0.53)	0.00 (±0.27)	0.00 (±0.09)	0.07 (±0.53)	0.07 (±0.53)	0.00 (±4.15)	0.00 (±1.38)
DBLP	0.00 (±0.23)	0.00 (±0.23)	0.00 (±3866.51)	0.00 (±69.47)	0.00 (±0.23)	0.00 (±0.23)	0.00 (±4.14)	0.00 (±1.38)
Full: bursty	12.69 (±13.64)	12.69 (±13.64)	0.00 (±0.00)	0.00 (±0.00)	12.69 (±13.64)	12.69 (±13.64)	0.00 (±0.00)	0.00 (±0.00)
Full: 1%	12.69 (±3.65)	12.69 (±3.64)	0.00 (±0.00)	0.00 (±0.00)	12.69 (±3.65)	12.69 (±3.64)	0.00 (±0.00)	0.00 (±0.00)
Full: 5%	10.67 (±7.91)	10.67 (±7.91)	0.00 (±0.00)	0.00 (±0.00)	10.67 (±7.91)	10.67 (±7.91)	0.00 (±0.00)	0.00 (±0.00)
Half: bursty	12.99 (±9.96)	12.99 (±9.97)	0.00 (±0.00)	0.00 (±0.00)	12.99 (±9.96)	12.99 (±9.97)	0.00 (±0.00)	0.00 (±0.00)
Half: 1%	12.99 (±4.49)	12.99 (±4.49)	0.00 (±0.00)	0.00 (±0.00)	12.99 (±4.49)	12.99 (±4.49)	0.00 (±0.00)	0.00 (±0.00)
Half: 5%	12.06 (±6.28)	12.06 (±6.29)	0.00 (±0.00)	0.00 (±0.00)	12.06 (±6.28)	12.06 (±6.29)	0.00 (±0.00)	0.00 (±0.00)

4.3.1 DBLP DBLP contains records of publications from the computer science domain, containing the title of the publication, authors, year of publication, and more. We perform a few case studies to highlight real world outliers found in the data.

Case Study 1. Pedro Olmo S. Vaz de Melo, Christos Faloutsos, and Antonio Alfredo Ferreira Loureiro. *Human Dynamics in Large Communication Networks.* SDM. 2011.

This is the first paper that we will look at that generated outliers (anomalous co-author pairs). This paper illustrates two interesting circumstances that can cause outliers, the bridging of fields of study and the overcoming of geographic boundaries. Christos Faloutsos primarily publishes in the fields of data mining and knowledge discovery, while, at the time of this publication, the first and third author primarily worked in the fields of distributed computing and wireless sensor networks. Moreover, Pedro Melo and Antonio Loureiro are researchers at universities in Brazil, while Christos Faloutsos is a professor at Carnegie Mellon in Philadelphia. So, how did this collaboration come to be? The lead author, Pedro Melo, was a visiting researcher for one year at Carnegie Mellon during his PhD. This paper is one of the first in what has become a regular collaboration among the authors, as shown by their numerous joint publications since this paper. Our algorithm successfully identified this new collaboration between Pedro and Christos and labeled it as an outlier.

Case Study 2. Charu C. Aggarwal, Yuchen Zhao, and Philip S. Yu. *A framework for clustering massive graph streams.* Statistical Analysis and Data Mining: The ASA Data Science Journal 3.6 (2010): 399-416.

Unlike Case Study 1, all of the authors from this paper live in the United States and have similar areas of research. This paper highlights a third cause of outliers; namely, the beginning of an author's career and collaborations. Prior to this paper, Charu Aggarwal and Philip Yu had a number of publications, including a co-authored book. At the time of this publication, however, Yuchen Zhao was a new PhD student at UIUC under Philip Yu. This was the first publication showing the collaboration of the authors. Our algorithm correctly labeled the edge between Charu and Yuchen as an outlier, and elucidated the beginning of Yuchen's publishing career and the beginning of his collaboration with Charu.

5 Related Work

Outlier detection over dynamic graphs has received an increasing amount of attention recently given its practical applications [18]. Traditionally, analysis is done by examining sequences of graph snapshots, where each snapshot captures the interactions for a given time window (e.g. 5 minutes of IP traffic). Snapshots are compared pairwise in chronological order (i.e. G_t is compared to G_{t-1}) and a scoring function maps the graph pair to a real number, creating a time series of scores. The scoring function can be based on community detection [8, 11, 12], graph distance [10, 17], tensor decomposition [20, 21], compression [19], or other graph features. A threshold is then applied to the time series of scores to identify events (isolated abnormalities), or change points (time points where significant changes occur in the graph and then persist). Aggarwal et. al. find dense patterns [3] and outliers [4] in graph streams, but assume that the size of any individual graph is extremely small compared to the universe it is drawn from. These small graphs are used as batch updates for the model of the stream.

Our method complements this existing body of work in a number of ways. We focus on the contrasts with [4], as it is the closest related work. In [4], Aggarwal et. al. propose a structural resevoir sampling algorithm that heuristically retains a subset of all edges seen in the graph stream. When a new graph arrives, the probability of each edge is calculated using the edges in the resevoir sample, and then aggregated into a single graph likelihood value. Graphs with a likelihood below a threshold value are output as outliers. Sampling enables

exact calculations to be performed, but relies on salient portions of the stream to be retained, with all other information being lost. Conversely, we maintain an approximate model of *entire* the edge stream, and thus use approximate calculations in our analysis. Using a sketch-based approximation enables a well defined (constant) upper bound on the space used, and we are able to prove probabilistic error bounds on the scoring functions used. Moreover, model updates and edge scoring are performed in constant time. Graph snapshot analysis typically perform more expensive operations, such as dynamically maintaining spanning forests [4], relying on the updates being less frequent, and thus are not well-suited for edge streams.

6 Conclusion

Outlier detection in dynamic networks has been a topic of growing interest in the past few years. While it has traditionally focused on graph streams, in this paper, we present the first approach for outlier detection in edge streams. Our sketch-based approach provides constant time updates and scoring functions, in addition to requiring constant space that is independent of the size of the edge stream and network. Experiments on synthetic datasets have shown that our approximations have only a marginal impact on the accuracy of the scoring functions, and we are still able to achieve high precision on identifying injected outliers, out performing the baseline method. Mining the DBLP co-authorship network elucidates some interesting types of outliers that exist, and our ability to capture them. In short, we have presented a scalable and effective method for outlier detection in edge streams.

7 Acknowledgements

This material is based upon work supported in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions or findings expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agent or entity of the US government. In addition, this material is based on work supported in part by the Department of Energy National Nuclear Security Administration under Award Number(s) DE-NA0002576.

References

[1] Enron network dataset – KONECT. `http://konect.uni-koblenz.de/networks/enron`. May, 2015.

[2] C. C. Aggarwal. *Data Streams: Models and Algorithms*, volume 31. Springer, 2007.

[3] C. C. Aggarwal, Y. Li, P. S. Yu, and R. Jin. On dense pattern mining in graph streams. *Proc. of the VLDB Endowment (PVLDB)*, 3(1):975–984, 2010.

[4] C. C. Aggarwal, Y. Zhao, and P. S. Yu. Outlier detection in graph streams. In *Proc. of the 27th Intl. Conf. on Data Engr. (ICDE)*, pages 399–409, 2011.

[5] L. Akoglu and C. Faloutsos. Rtg: A recursive realistic graph generator using random typing. *Data Mining and Knowledge Discovery*, 19(2):194–209, 2009.

[6] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2014.

[7] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[8] Z. Chen, W. Hendrix, and N. F. Samatova. Community-based anomaly detection in evolutionary networks. *J. Intelligent Information Systems*, 39(1):59–85, 2012.

[9] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[10] C. Faloutsos, D. Koutra, and J. T. Vogelstein. DELTACON: A principled massive-graph similarity function. In *Proc. of the 13th SIAM Intl. Conf. on Data Mining*, pages 162–170, 2013.

[11] M. Gupta, J. Gao, Y. Sun, and J. Han. Integrating community matching and outlier detection for mining evolutionary community outliers. In *Proc. of the 18th ACM SIGKDD Conf.*, pages 859–867, 2012.

[12] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova. Community detection in large-scale networks: A survey and empirical evaluation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2014.

[13] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[14] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. American Soc. for Information Sci. and Tech.*, 58(7):1019–1031, 2007.

[15] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[16] S. Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[17] B. Pincombe. Anomaly detection in time series of graphs using ARMA processes. *ASOR Bulletin*, 24(4):2, 2005.

[18] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova. Anomaly detection in dynamic detworks: A survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.

[19] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: Parameter-free mining of large time-evolving graphs. In *Proc. of the 13th ACM SIGKDD Conf.*, pages 687–696, 2007.

[20] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proc. of the 12th ACM SIGKDD Conf.*, pages 374–383, 2006.

[21] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos. Colibri: fast mining of large static and dynamic graphs. In *Proc. of the 14th ACM SIGKDD Conf.*, pages 686–694, 2008.

Table 3: Notations

Symbol	Meaning
ε	Error factor for query response from sketch.
δ	Probability of count-query response from the sketch being within a factor of ε is at least $1-\delta$.
w	Width of a sketch. $w = \lceil \frac{e}{\varepsilon} \rceil$.
d	Depth of a sketch. $d = \lceil \ln \frac{1}{\delta} \rceil$.
$w(u,v)$	Weight of the edge between vertices u and v. Equal to the number of times (u,v) has appeared in the stream.
$N(u)$	Neighborhood of u. $N(u) = \{v \mid (u,v) \in \mathcal{E}\}$.
N_u	Sketch for vertex u (used for estimating the homophily score).
N_{uv}	Number of cells that have a nonzero value in *both* N_u and N_v.
$w(u)$	Weight of a vertex u, defined as the sum of the weight of its edges. $w(u) = \sum_{k \in N(u)} w(u,k)$.
$\lvert\mathcal{E}\rvert$	Total number of edges seen in the stream.
$\lvert V\rvert$	Number of unique vertices seen in the stream.
s^s_{uv}	Sample score of edge (u,v). Defined by Eqn. 2.1.
s^p_{uv}	Preferential attachment score of edge (u,v). Defined by Eqn. 2.2.
s^h_{uv}	Homophily score of edge (u,v). Defined by Eqn. 2.3.
s_{uv}	Total score of an edge (u,v). Defined by Eqn. 2.4.
$h_i(u)$	Value of the ith hash function applied to u.

A Proofs

LEMMA A.1. *Using a Count-Min sketch of width* $w = \lceil \frac{e}{\varepsilon} \rceil$ *and depth* $d = \lceil \ln \frac{1}{\delta} \rceil$ *the estimated sample score of an edge* (u,v), s^{se}_{uv}, *is bound by the following inequality with probability* at least $1-\delta$.

$$s^s_{uv} \leq s^{se}_{uv} \leq s^s_{uv} + \varepsilon$$

Proof. **Lower bound:** The exact sample score of an edge (u,v) is defined as $s^s_{uv} = w(u,v)/|\mathcal{E}|$. The estimated sample score is then given by the equation $s^{se}_{uv} = w_e(u,v)/|\mathcal{E}|$, where $w_e(u,v)$ is the estimated weight of the edge using the Count-min sketch. As the estimate of the sketch is guaranteed to be greater than or equal to the true value, clearly, $s^s_{uv} \leq s^{se}_{uv}$.

Upper bound: To prove the upper bound, it is sufficient to show that the following inequality holds with probability at least $1-\delta$.

$$w_e(u,v) \leq w(u,v) + |\mathcal{E}|\varepsilon \tag{A.1}$$

For a single hash function, the estimated value of an edge weight is equal to $w_e(u,v) = w(u,v) + X_i$, where i is the index that (u,v) hashes to and X_i is a variable accounting for the error produced by hash collisions in cell i. Assuming a pairwise independent hash function, the expected value of the error for any given edge weight is $\mathbb{E}[X_i] \leq |\mathcal{E}|/w = m_t\varepsilon/e$. By the Markov inequality,

$$\begin{aligned}\mathbb{P}[w_e(u,v) &> w(u,v) + |\mathcal{E}|\varepsilon] \\ &= \mathbb{P}[w(u,v) + X_i > w(u,v) + |\mathcal{E}|\varepsilon] \\ &= \mathbb{P}[X_i > e\mathbb{E}[X_i]] < \frac{\mathbb{E}[X_i]}{e\mathbb{E}[X_i]} = e^{-1}\end{aligned}$$

From this, we can see that for d independent hash functions, $\mathbb{P}[w_e(u,v) > w(u,v) + |\mathcal{E}|\varepsilon] < e^{-d} \leq \delta$. Thus, Eqn. A.1 holds with probability at least $1-\delta$.

LEMMA A.2. *Using a Count-Min sketch of width* $w = \lceil \frac{e}{\varepsilon} \rceil$ *and depth* $d = \lceil \ln \frac{1}{\delta} \rceil$, *the estimated preferential attachment score of an edge* (u,v), s^{pe}_{uv}, *is bound by the following inequality with probability* at least $1-\delta$.

$$s^p_{uv} \leq s^{pe}_{uv} \leq s^p_{uv} + \varepsilon$$

Proof. **Lower bound:** The exact preferential attachment score of an edge is defined as $s^p_{uv} = w(u)w(v)/|\mathcal{E}|^2$ for some edge (u,v). The estimated preferential attachment score is then given by the equation $s^{pe}_{uv} = w_e(u)w_e(v)/|\mathcal{E}|^2$, where $w_e(u)$ is the estimated score (using the Count-min sketch) for vertex u. As the estimates of the sketch are guaranteed to be greater than or equal to the true value, clearly, $s^p_{uv} \leq s^{pe}_{uv}$.

Upper bound: To prove the upper bound, it is sufficient to show that $w_e(u)w_e(v) - w(u)w(v) \leq |\mathcal{E}|^2\varepsilon$ with probability at least $1-\delta$. Let $h(v)$ denote the hash function h applied to vertex v. Observe that, for a single hash function,

$$w_e(u)w_e(v) = \left(w(u) + \sum_{\substack{p \neq u,\\ h(p)=h(u)}} w(p)\right)\left(w(v) + \sum_{\substack{p \neq v,\\ h(p)=h(v)}} w(p)\right)$$

Then, assuming a pairwise independent hash function (with sketch width w), $\mathbb{E}[w_e(u)w_e(v) - w(u)w(v)] \leq |\mathcal{E}|^2\varepsilon/e$ (see Figure 3 for the derivation): By the Markov inequality, we know that $\mathbb{P}[w_e(u)w_e(v) - w(u)w(v) > |\mathcal{E}|^2\varepsilon] < e^{-1}$. Hence, for d independent hash functions, $\mathbb{P}[w_e(u)w_e(v) - w(u)w(v) > |\mathcal{E}|^2\varepsilon] < e^{-d} \leq \delta$.

$$\mathbb{E}[w_e(u)w_e(v) - w(u)w(v)] = \left(w(u) + \frac{1}{w}\sum_{p\neq u} w(p)\right)\left(w(v) + \frac{1}{w}\sum_{p\neq v} w(p)\right) - w(u)w(v)$$
$$= \frac{1}{w}\left(w(u)\sum_{p\neq v} w(p) + w(v)\sum_{p\neq u} w(p) + \frac{1}{w}\sum_{p\neq u}\sum_{q\neq v} w(p)w(q)\right) \leq \frac{|\mathcal{E}|^2}{w} = \frac{|\mathcal{E}|^2\varepsilon}{e}$$

Figure 3: Derivation for an upperbound of $\mathbb{E}[w_e(u)w_e(v) - w(u)w(v)]$

LEMMA A.3. *Assume every vertex has their own sketch to track the weight of the edges between themselves and every vertex in their neighborhood. Given two vertices, u and v, and their corresponding sketches, N_u and N_v, each of width $w = \lceil \frac{e}{\varepsilon} \rceil$ and depth $d = \lceil \ln \frac{1}{\delta} \rceil$, the estimated homophily score of an edge (u, v), s^{he}_{uv}, is bound by the following inequality with probability* at least $1 - \delta$,

$$s^h_{uv} \leq s^{he}_{uv} \leq s^h_{uv} + \varepsilon N_{uv} \tag{A.2}$$

where N_{uv} is the number of cells that have a nonzero value in both N_u and N_v.

Proof. **Lower bound:** The exact and estimated homophily scores defined, respectively, as:

$$s^h_{uv} = \frac{\sum_{k\in N(u)\cap N(v)}(w(u,k) + w(v,k))}{\sum_{k\in N(u)} w(u,k) + \sum_{k\in N(v)} w(v,k)}, \text{ and}$$
$$s^{he}_{uv} = \frac{\sum_{k\in N(u)\cap N(v)}(w_e(u,k) + w_e(v,k))}{\sum_{k\in N(u)} w_e(u,k) + \sum_{k\in N(v)} w_e(v,k)}$$

Notice that the denominators in the above equations are equivalent because the sum of any row in the sketch for a vertex is the exact weight of that vertex; namely, $||N_u|| = \sum_{k\in N(u)} w(u,k)$. As a result, we get the following:

$$s^h_{uv} = \frac{\sum_{k\in N(u)\cap N(v)}(w(u,k) + w(v,k))}{||N_u|| + ||N_v||}, \text{ and}$$
$$s^{he}_{uv} = \frac{\sum_{k\in N(u)\cap N(v)}(w_e(u,k) + w_e(v,k))}{||N_u|| + ||N_v||}$$

Assuming every vertex's sketch uses the same set of hash functions, the estimated weight of the intersection can be broken into two components, the true weight of the intersection plus the error introducted by hash collisions.

$$\sum_{k\in N(u)\cap N(v)}(w_e(u,k) + w_e(v,k)) = \sum_{k\in N(u)\cap N(v)}(w(u,k) + w(v,k)) + \sum_{\substack{q\notin N(u)\cap N(v),\\ \exists p\in N(u)\cap N(v), h(p)=h(q)}}(w(u,q) + w(v,q)) \tag{A.3}$$

It is clear from this that the estimated score will always be at least the non-estimated score.

Upper bound: We will show that $s^{he}_{uv} \leq s^h_{uv}$ holds with probability at least $1 - \delta$ by showing that the following inequality holds with probability at least $1-\delta$. It is clear that if we can show that

$$\sum_{k\in N(u)\cap N(v)}(w_e(u,k) + w_e(v,k)) - \sum_{k\in N(u)\cap N(v)}(w(u,k) + w(v,k)) \leq N\varepsilon(||N_u|| + ||N_v||). \tag{A.4}$$

holds with probability at least $1-\delta$, then Eqn. A.2 does as well.

Let X be a variable defined as the left hand side of Eqn. A.4 and $p = \mathbb{P}[\exists p \in N(u) \cap N(v), h(p) = h(q)]$. Substituting Eqn. A.3 into Eqn. A.4 and simplifying we get:

$$\mathbb{E}[X] = \sum_{q\notin N(u)\cap N(v)} p(w(u,q) + w(v,q))$$
$$\leq \sum_{q\notin N(u)\cap N(v)} \frac{N_{uv}}{w}(w(u,q) + w(v,q))$$
$$\leq \frac{N_{uv}}{w}(||N_u|| + ||N_v||) = \frac{\varepsilon N_{uv}}{e}(||N_u|| + ||N_v||)$$

By the Markov inequality, $\mathbb{P}[X > N_{uv}\varepsilon(||N_u|| + ||N_v||)] \leq e^{-1}$. Generalizing to d different hash functions gives the desired result.

THEOREM A.1. *Using a CM sketch of width $w = \lceil \frac{e}{\varepsilon_1} \rceil$ and depth $d = \lceil \ln \frac{1}{\delta} \rceil$ for estimating sample scores, a CM sketch of $w = \lceil \frac{e}{\varepsilon_2} \rceil$ and $d = \lceil \ln \frac{1}{\delta} \rceil$ for estimating preferential attachment scores, and $|V|$ CM sketches of $w = \lceil \frac{e}{\varepsilon_3} \rceil$ and $d = \lceil \ln \frac{1}{\delta} \rceil$ for estimating homophily scores, the total estimated score of an edge (u, v), s^e_{uv}, is bound by the following inequality with probability at least $1 - \delta$.*

$$s_{uv} \leq s^e_{uv} \leq s_{uv} + \alpha\varepsilon_1 + \beta\varepsilon_2 + \gamma\varepsilon_3 N_{uv} \tag{A.5}$$

where α, β, and γ are weighting coefficients such that $\alpha + \beta + \gamma = 1$.

Proof. It follows from the linearity of expected values and Lemmas A.1, A.2, and A.3.

B Complexity Analysis

Let w_1 (w_2) be the width of the sketches used for storing vertex (edge) weights, w_3 be the width of the sketches storing vertex neighborhood information, d be the depth of all the sketches, and l be the number of sketches used in the shared sketch heuristic ($l = |V|$ in the individual sketch approximation). Each edge in the stream is processed in $O(w_3 d)$ time, and the total space required is $O(w_1 d + w_2 d + w_3 dl)$.

B.1 Time The run time is determined by the operations required to update the sketches and score an edge.

Model Updates. When a new edge is added to the model, all of the corresponding weights must be updated. To update the weight of an edge (vertex), all of the d cells in the sketch corresponding the edge (vertex) must be incremented, requiring d hashes, and thus $O(d)$ time. Updating the neighborhood information for the two vertices incident on the edge requires $2d$ hashes, plus an additional two for the shared sketch heuristic, and is thus $O(d)$. Model updates are therefore $O(d)$.

Edge scoring. The sample (preferential attachment) score requires one (two) normalized weight estimations. Similar to updating the weights, querying a weight requires $O(d)$ time, hence the sample and preferential attachment scoring functions are $O(d)$. Calculating the homophily score of an edge (u, v) requires traversing two sketches, $O(w_3 d)$, and normalizing by two weight estimations, $O(d)$. Therefore, scoring an edge is $O(wd)$.

Total. Processing each edge in the stream then requires $O(d + w_3 d) = O(w_3 d)$ time.

B.2 Space The required space is proportional to the size of the sketches used, and the number of sketches used. The vertex and edge sketches require $O(w_1 d)$ and $O(w_2 d)$ space, respectively. Approximating neighborhood information requires l sketches, and therefore $O(w_3 dl)$ space. In total, $O(w_1 d + w_2 d + w_3 dl)$ space is required.

C Experimental Setup

All experiments were performed on a dedicated Intel server consisting of two hex-core E5645 processors and 64GB DDR2 RAM. The data was stored on a 2TB RAID1 partition, and the operating system was installed on a 120.5GB SSD. The approximate and heuristic versions are written in C++, and the exact version is written in MATLAB. An exact version was also written in C++, however, it ran significantly slower than the equivalent version in MATLAB due to the handling of sparse matrix operations, so the MATLAB code was used for all experiments. The C++ code was compiled using GCC 4.8.2 with no optimization flags.

D Dataset Descrptions

D.1 Enron The original Enron email dataset was downloaded from `https://www.cs.cmu.edu/~./enron/` and processed into an edgelist. All emails associated with the Enron domain that were in the To, From, CC, or BCC fields of the emails were considered for creating edges. One email generated an edge for each unique (sender, receiver) pair. The edge list is chronologically ordered, but for a single email the ordering of the (sender, receiver) pairs generated is random.

In total, six types of anomalous edge lists were created. Three had outliers injected throughout the entire stream, while the other three had outliers only in the second half of the stream. Each of these two categories contained the same three types of injections: (1) *1% outliers.* For every edge in the stream, there was a 1% chance to randomly inject an outlier (i.e., anomalous edge); (2) *5% outliers.* For every edge in the stream, there was a 5% chance to randomly inject an outlier; (3) *1% bursty outliers.* For every edge in the stream, there is a .1% chance to inject a group of 10 outliers.

D.2 DBLP DBLP contains records of publications from the computer science domain, containing the title of the publication, authors, year of publication, and more. The original data was downloaded from `http://dblp.uni-trier.de/xml/` and processed into an edgelist of co-author pairs. All conference papers, journal articles, and books from 1954 to June 10, 2015 were used. We process the data and create a timestamped author pair edge list stream by generating an edge for every author pair on every paper. For example, if a paper published had 3 authors, then $\binom{3}{2}$ edges are generated. As only the year of publication is available, the stream is processed chronologically and then alphabetically within each year.

D.3 IMDB IMDB contains information about movies and television shows from the past few decades. The original data was downloaded from `http://www.imdb.com/interfaces` and processed into an edgelist where actors are vertices and an edge connects two actors if they co-star in a movie or TV show. All productions from 1970 until 2006 were used.

E Note on Neighborhood Intersections

There is a minor but important distinction between using sketches and using an adjacency matrix to calculate the weight of a neighborhood intersection. With sketches, d estimates are calculated for the intersection weight, where the ith estimate is the sum of the cells

that are nonzero in the ith row of both sketches. Then, the estimated weight of the neighborhood intersection is the minimum of the d estimates. In addition to minimizing the individual edge weight errors, this minimizes the weight of false positives that occur when vertices $u \in N(u) \setminus N(v)$ and $v \in N(v) \setminus N(u)$ hash to the same cell in the sketch.