Mining Persistent Activity in Continually Evolving Networks

Caleb Belth  
University of Michigan  
cbelth@umich.edu

Xinyi Zheng  
University of Michigan  
zxycharl@umich.edu

Danai Koutra  
University of Michigan  
dkoutra@umich.edu

ABSTRACT

Frequent pattern mining is a key area of study that gives insights into the structure and dynamics of evolving networks, such as social or road networks. However, not only does a network evolve, but often the way that it evolves, itself evolves. Thus, knowing, in addition to patterns’ frequencies, for how long and how regularly they have occurred—i.e., their persistence—can add to our understanding of evolving networks. In this work, we propose the problem of mining activity that persists through time in continually evolving networks—i.e., activity that repeatedly and consistently occurs. We extend the notion of temporal motifs to capture activity among specific nodes, in what we call activity snippets, which are small sequences of edge-updates that reoccur. We propose axioms and properties that a measure of persistence should satisfy, and develop such a persistence measure. We also propose PENminer, an efficient framework for mining activity snippets’ persistence in Evolving Networks, and design both offline and streaming algorithms. We apply PENminer to numerous real, large-scale evolving networks and edge streams, and find activity that is surprisingly regular over a long period of time, but too infrequent to be discovered by aggregate count alone, and bursts of activity exposed by their lack of persistence. Our findings with PENminer include neighborhoods in NYC where taxi traffic persisted through Hurricane Sandy; the opening of new bike-stations, characteristics of social network users, and more. Moreover, we use PENminer towards identifying anomalies in multiple networks, outperforming baselines at identifying subtle anomalies by 9.8-48% in AUC.

ACM Reference Format:

1 INTRODUCTION

Many networks evolve continually over time, such as traffic networks that encode current en-route traffic, networks of computers (IP-addresses) sending messages to each other, social networks of users interacting over time, and more. In order to understand these networks and the systems they represent, it is important to consider not just their structure, but also their temporal dynamics. Towards this goal, existing works have focused on counting patterns in networks. These patterns include temporal motifs (small subgraph patterns) [14, 17, 21, 24], frequent subgraphs in evolving networks [3, 4, 6], flow motifs [16], communication motifs [30], and coevolving relational motifs [5]. This line of research uses the frequency of patterns towards understanding the behavior of networks (e.g., some interaction patterns are more common in Q&A forums than in instant messengers [24]). However, many patterns in evolving networks may only last for a short period of time (e.g., bursty activity). This can lead to large aggregate counts, making a burst anomaly falsely appear to be an intrinsic characteristic of a dynamic network. On the other hand, there may be important activity in a network that has low overall frequency, but that occurs continually and regularly [27], such as a stealthy computer-network attack. Thus, to more fully understand the dynamics of evolving networks, a pattern’s persistence—involving how long it has occurred, how uniformly its occurrences are spread out, and what its frequency is—should be taken into account (Fig. 1).

In this work, we seek to mine persistent activity in continually evolving networks. We view the activity in a network as a stream of edge updates (insertions or deletions) over time. We introduce activity snippets by extending the idea of temporal motifs [24], which encode activity among nodes in general, to allow for encoding activity between or among specific nodes. This extension allows us to capture exact activity, (e.g., a specific edge in the network), which is important for applications like identifying which public transportation routes are used persistently, or identifying suspicious nodes or edges (e.g., nodes 4 and 5 in Fig. 1). In other applications, such as social network analysis, we may be interested in interactions between nodes in general, with no specific users in mind. In this case, activity snippets reduce to temporal motifs. In either case, activity can repeat, whether it be specific nodes repeatedly engaging, or different sets of nodes behaving in consistent ways. We seek to analyze this activity in terms of how persistently it occurs.

Our main contributions are as follows:
• Precise Formulation of Persistence: We introduce four axioms and three general properties that a measure of persistence ought to satisfy. We also provide a versatile persistence measure, and prove that it satisfies all axioms and properties.

• Offline and Online Algorithms: We develop PENminer, an efficient framework that uses our persistence measure to mine evolving networks. Our offline version, sPENminer, uses the measure for analyzing time-stamped sequences of edges from the past. Our online version, sPENminer, computes the measure incrementally for real-time analysis of edge-streams.

• Extensive Experiments on Real Data: We perform experiments on real, large scale evolving networks, which reveal real-world phenomena, including neighborhoods in NYC where taxi traffic persisted through Hurricane Sandy, the opening of new bike-stations in multiple cities, characteristics of social network users, and more. We also demonstrate that PENminer is scalable in offline and streaming settings, processing edge-updates in each stream 10K to 360K times faster than the rate of that stream. This allows both subtle and bursty activity to be identified in real-time, right when it happens. PENminer also effectively identifies subtle anomalies, outperforming baselines by 9.8-48% in AUC.

Our code is available at https://github.com/GemsLab/PENminer.

2 RELATED WORK

Our work is related to motif mining, frequent subgraph mining, and persistent item mining.

Motif Mining. Network motifs, which are small, frequent subgraph patterns [23] are used in various problems, such as network summarization [22] and exploration, and dense subgraph detection [12]. Temporal motifs [17] extend static motifs to evolving networks by adding an ordering to their edges. Researchers have analyzed temporal motifs within and across a wide-range of networks [24], and have proposed sampling methods to estimate their counts [21]. Temporal motifs have also been extended to capture the flow of information among nodes in a motif [16]. Among different motifs, triangle counting has attracted significant interest [28].

A special case, communication motifs, was introduced to model temporal communication patterns of users in communication networks, for the purpose of understanding information flow [14, 30]. In [30], focusing on motif counts, the authors observed stability in the ranking of the ten most frequent motifs over time, but did not investigate subtly persistent or bursty behavior. Coevolving Relational Motifs (CRMs) are patterns that describe sets of nodes that evolve together in a consistent way [5]. In CRMs, consistency means that when a set of nodes co-evolve, it is almost always in the way specified by the motif. Thus, it considers the relative frequency of the motif, compared to other possible motifs over the same nodes, but does not investigate persistence.

Our work differs from motif-mining in its focus on persistence not just counts, and much of our analysis deals with specific edges or sequences of edges, rather than general activity among nodes.

Frequent Subgraph Mining. Frequent subgraph mining (FSM) has two settings: transactional FSM and single graph FSM [15]. The transactional setting attempts to identify, in a sequence of graphs (transactions), subgraphs that appear in a large number of the graphs. This setting naturally extends to evolving networks, by treating graph snapshots (corresponding to a batch of edge updates) as transactions [25]. The single graph setting seeks to identify subgraphs that have many instances within a single, long graph [10]. It has also been extended to evolving networks, including edge streams [3, 4, 6], where the goal is to adaptively maintain the most frequent subgraphs. While these methods seek to find frequent subgraphs, we seek to find persistent activity.

Persistent Item Mining. Motivated by scenarios of stealthy click-fraud or distributed denial-of-service attacks, persistent item mining in data streams was introduced in [9]. Arguing that, besides frequency, the length of the time period in which an item appears is important for understanding the dynamics of streaming data, the authors introduced a heuristic definition of a persistent item as one that occurs at least once in a large number of equally-sized observation periods. As we show, this simple definition violates some intuitive desired properties (§ 3.4). Our work goes beyond heuristics to establish a technical definition of a persistence measure, and focuses on edge-streams.

Persistent community detection, which attempts to find communities that last for long durations of time, has also been studied [19, 26]. However, these works seek to find tightly-knit subgraphs (communities) that last for a long time, while we focus on measuring activity that regularly reoccurs through time without requiring that this activity take place in contiguous chunks of time.

3 THEORY

3.1 Preliminary Definitions

We begin with preliminary definitions. Since most definitions apply beyond network settings, we first introduce the general concepts in § 3.1.1 and then the network-specific terminology in § 3.1.2.

3.1.1 Events in Time. Interval of Time. An interval of time $[t_s, t_e]$ (Fig. 2) is defined as $\{t : t \in \mathbb{R}_{\geq 0}, t \in [t_s, t_e]\}$. The width of the interval is $\|t_s, t_e\| = t_e - t_s \geq 0$. The set of all intervals is $I \triangleq \{[t_s, t_e] : t_s, t_e \in \mathbb{R}_{\geq 0}, t_s \leq t_e\}$.

Event. An event $x$, with respect to an interval $[t_s, t_e]$, is something that occurs at least once in that interval. Examples include item purchases in a sequence of transactions, and measurements in a time series. The universe of events is $X$.

Event Occurrence. An occurrence, $t \in [t_s, t_e]$ of $x$, is a timestamp. All the occurrences of $x$, $O_x = \{t_f, \ldots, t_l\}$, form an ordered multiset of timestamps between the first and last occurrences, and the corresponding interval of occurrences is $[t_f, t_l] \subseteq [t_s, t_e]$ (e.g., Fig. 2). We denote the ordered set of unique occurrences $\hat{O}_x$. 

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{F}$</td>
<td>Edge stream</td>
</tr>
<tr>
<td>$x$</td>
<td>Activity snippet</td>
</tr>
<tr>
<td>$k_{max}, \delta_{max}$</td>
<td>Maximum size and duration for a snippet</td>
</tr>
<tr>
<td>$\phi$</td>
<td>View of a snippet (ID, LABEL, or ORDER)</td>
</tr>
<tr>
<td>$[t_f, t_l], |t_j, t_l|$</td>
<td>Interval of time and its width $t_l - t_j$</td>
</tr>
<tr>
<td>$O_x, \hat{O}_x$</td>
<td>All and unique occurrences of $x$ in $[t_s, t_e]$, resp.</td>
</tr>
<tr>
<td>$\Gamma_x$</td>
<td>Gaps (lengths of time) between unique occurrences of $x$</td>
</tr>
</tbody>
</table>

Table 1: Description of major symbols.
Occurrence Gaps. The gaps (i.e., amount of time) between occurrences of x form the sequence \( G_x = (g_1, \ldots, g_{|O_x|-1}) \), where \( g_i = t_{i+1} - t_i \) for \( t_i, t_{i+1} \in \tilde{O}_x \). The number of gaps is \( |G_x| = |\tilde{O}_x| - 1 \).

3.1.2 Activity Snippets in Evolving Networks. Graph or Network. A graph or network \( G = (V, E) \) is a set of nodes \( V \), and a set of edges \( E \subseteq V \times V \). If an edge between nodes \( v_1 \) and \( v_2 \) has a relationship type \( r \), we denote it \( (v_1, r, v_2) \). Our proposed persistence measure and algorithms apply to general networks: labeled (nodes/edges), directed, weighted, bipartite or multi-graphs.

Edge-update. An edge update \( u = (x, v_1, r, v_2, t) \) to a network \( G \) is the insertion (+) or deletion (-) of an edge \( (v_1, r, v_2) \) at time \( t \). We refer to \( y \)'s timestamp with \( x \).

Edge Stream. An edge stream \( \mathcal{S} = (u_1, u_2, \ldots) \) is a time-ordered sequence of possibly infinite edge-updates to an evolving graph \( G = (G_1, G_2, \ldots) \). We denote the start time of \( \mathcal{S} \), \( \text{start}(\mathcal{S}) = \text{timestamp}(u_1) \) its length \( |\mathcal{S}| \), and if bounded, its end time \( \text{end}(\mathcal{S}) = \text{timestamp}(u_n) \). We call a sub-sequence of \( \mathcal{S} \) that consists of all edge-updates in the last \( w \) time units a window \( \mathcal{W} \) of width \( w \).

Activity Snippet. Intuitively, an activity snippet \( x \) describes a sequence of activity among connected nodes in the network. Specifically, \( x = (\phi(u_1), \ldots, \phi(u_n)) \) is a sequence of ordered, but not necessarily contiguous, edge updates, where the node IDs may be replaced by a view \( \phi \): (1) their labels or (2) the position of their first occurrence in the sequence (Fig. 3). If the node IDs are not replaced, the snippet is an exact sequence of activity. If the IDs are replaced by their position in the sequence, then the snippet is analogous to temporal motifs [24], but also captures edge deletions or insertions. The snippet has duration \( \delta = \text{timestamp}(u_n) - \text{timestamp}(u_1) \) and size \( k \) equal to the number of edge-updates in the snippet. A \((\delta_{\text{max}}, k_{\text{max}})\) activity snippet is an activity snippet with duration \( \delta \leq \delta_{\text{max}} \) and size \( k \leq k_{\text{max}} \). In this work, events (§ 3.1.1) are activity snippets.

Problem Definition. Given these definitions, we focus on the problem of persistent activity mining:

Problem 1 (Persistent Activity Mining). Given a network \( G \) that continually evolves via an edge-stream \( \mathcal{S} \), measure the persistence of each activity snippet \( x \), i.e., for how long, how often, and how regularly it has occurred.

3.2 Properties of Persistence

Based on our definitions, we present axioms and properties that a persistence measure should follow. In the remaining sections, we propose a principled persistence measure (§ 3.3) and prove that it satisfies all the axioms and properties (§ 3.4). Although our theoretical definitions, properties, proposed measure and derivations are general and apply to any event \( x \) in a stream or time series, in the context of Problem 1, \( x \) is an activity snippet.

DEFINITION 1 (Persistence Measure). A persistence measure \( P(x; [t_s, t_e]) : X \times \mathcal{I} \rightarrow \mathbb{R}_{\geq 0} \) is a function that defines the persistence of an event \( x \in X \) in the interval \( [t_s, t_e] \in \mathcal{I} \), and satisfies the following axioms:

- **A1**: It is non-negative, and 0 if there are no occurrences. Formally, \( P(x; [t_s, t_e]) \geq 0 \), with equality iff \( |O_x| = 0 \).
- **A2**: As the interval becomes infinitely filled with unique occurrences, persistence tends to infinity. That is, \( \lim_{|\mathcal{O}_x| \rightarrow \infty} P(x; [t_s, t_e]) = \infty \).
- **A3**: Shifting all occurrences in time does not affect persistence. Formally, \( P(x'; [t_s, t_e]) = P(x; [t_s, t_e]) \), where \( x' \) is an event with occurrences \( O_{x'} = \{ t + c : t \in O_x \} \), for some constant \( c \). Such that the shifted points remain in \( [t_s, t_e] \), i.e., \( t_f + c \geq t_s, t + c \leq t_e \).

Besides these axioms, there are several properties that a good persistence measures ought to follow. For use in presenting these properties, let \( x \) a special class of event with \( n \) occurrences, all of which are unique and uniformly spaced over \( [t_f, t_l] \), that is, \( |O_{x^n}| = |O_{x^n}| = n \) and \( g_i = g_j = \frac{|[t_f, t_l]|}{|O_{x^n}|} \), \( \forall g_i, g_j \in \Gamma_{x^n} \).

**P1**: For two events with \( n \) unique, uniformly-spaced occurrences, persistence is larger for the event with occurrences spread over a wider interval. Formally, for any \( x_{1}^{n} \) and \( x_{2}^{n} \) such that \( |[t_f, t_l]|_{x_{1}} < |[t_f, t_l]|_{x_{2}} \), it should hold that \( P(x_{1}^{n}; [t_s, t_e]) < P(x_{2}^{n}; [t_s, t_e]) \).

**P2**: For two events with unique, uniformly-spaced occurrences spread over the same interval, persistence is larger for the event with more occurrences. Mathematically, for any \( x_{1}^{n} \) and \( x_{2}^{n} \) such that \( |[t_f, t_l]|_{x_{1}} = |[t_f, t_l]|_{x_{2}} \), the persistence measure should satisfy \( P(x_{1}^{n}; [t_s, t_e]) < P(x_{2}^{n}; [t_s, t_e]) \), \( \forall k \geq 1 \).

**P3**: The persistence of an event with \( n \) unique occurrences in an interval is maximized iff the occurrences are spread out uniformly. Formally, for any \( x \) and \( x^{n} \) such that \( |O_x| = |O_{x^n}| = n \) and \( |[t_f, t_l]|_x = |[t_f, t_l]|_{x^n} \), the persistence measure should satisfy \( P(x; [t_s, t_e]) \leq P(x^{n}; [t_s, t_e]), \) with equality iff \( \Gamma_x = \Gamma_{x^n}, \forall n > 2 \).

3.3 Proposed Persistence Measure

Family of Persistence Measures. The axioms and properties point to three main components of persistence: the width of the interval in which occurrences fall, the frequency, and the distribution of occurrences. We thus propose a family of persistence measures

\[
P(x; [t_s, t_e]) \triangleq f \left( W(x; [t_s, t_e]), F(x; [t_s, t_e]), S(x; [t_s, t_e]) \right)
\]

where \( W(\cdot) \) is a function of the width of \( x \)'s interval of occurrences \( [t_f, t_l] \), \( F(\cdot) \) is a function of the number of occurrences, \( S(\cdot) \) is a function of how uniformly the points are spread out over \( [t_f, t_l] \), and \( f(\cdot) \) is a function that combines these components.
**A Persistence Measure.** There are a number of ways to construct the functions $W(\cdot), F(\cdot), \text{and } S(\cdot)$, and to combine them. We propose one intuitive persistence measure within the family described above and show that it satisfies all the axioms and properties (§ 3.4).

First, we define $W(\cdot)$ as the percentage of the interval width that is covered by the occurrences of $x$:

$$W(x;[t_s,t_e]) \triangleq \frac{|\{t_f,t_j\}| + 1}{|t_s,t_e|} + 1$$

where we add one in the numerator and denominator so that they are non-zero when $t_f = t_j$ or $t_s = t_e$. For $|O_x| = 0$, we define $W(\cdot) = 0$.

Second, we define $F(\cdot)$, the function of frequency, as the logarithm of the number of occurrences $|O_x|$ to: (a) prevent the term from dominating over the others (which can be more than an order of magnitude smaller), and (b) naturally capture diminishing returns. Formally,

$$F(x;[t_s,t_e]) \triangleq \log_{10}(|O_x| + 1),$$

where we add one to ensure that the logarithm is well-defined in the absence of occurrences.

Third, for $S(\cdot)$, to capture how regularly the occurrences are spread, we model the distribution of the gaps $\Gamma_x = (g_1, \ldots, g_i, \ldots)$ in a principled way via Shannon entropy [8]. Entropy measures the amount of randomness in a distribution, and when the distribution has a fixed number of outcomes in its support, this essentially means it measures the distribution’s uniformity. The gaps $\Gamma_x$ between occurrences (§ 3.1.1) normalized by the interval width $|\{t_f,t_j\}|$ define a probability mass function with entropy

$$H(\Gamma_x) \triangleq -\sum_{i=1}^{\infty} g_i \cdot \log_{10}(g_i)$$

where we add one since entropy is 0 if there are 0 or 1 gaps.

Since the terms defined above have different units, we combine them into one function as follows:

$$P(x;[t_s,t_e]) \triangleq W(x;[t_s,t_e])^\alpha F(x;[t_s,t_e])^\beta S(x;[t_s,t_e])^\gamma$$

where the finite exponents $\alpha, \beta, \gamma \in (0, \infty)$ can be used for assigning different weights to the components, depending on the goals of a particular application (e.g., $\gamma > 1$ can help events with low frequency but high regularity to be discovered).

### 3.4 Theoretical Analysis

We now show that Eq. (6) is a principled persistence measure that satisfies all axioms and properties. We also discuss a recently-proposed persistence heuristic, which violates key axioms.

**Lemma 1.** If $x$ has occurred at least once, $|O_x| > 0$, then our proposed measure is positive: $P(x;[t_s,t_e]) > 0$, $\forall \alpha, \beta, \gamma \in (0, \infty)$.

**Proof.** $W(x;[t_s,t_e]) > 0$ since $|\{t_f,t_j\}| + 1 \geq |\{t_f,t_j\}| + 1 > 0$. Since $|O_x| > 0$, $F(x;[t_s,t_e]) > 0$. If $|O_x| \in [1,2]$, then $|\Gamma_x| \in [0,1]$, so $S(x;[t_s,t_e]) = 1 > 0$ by definition. If $|O_x| > 2$, log $|\Gamma_x| > 0$ and the well-known property of entropy, $H(\Gamma_x) \geq 0$ [8], implies $S(x;[t_s,t_e]) > 0$. For $\alpha, \beta, \gamma > 0$, each component is still greater than 0 and so is their product $P(x;[t_s,t_e])$.

**Theorem 1.** $P(x;[t_s,t_e])$ as defined in Eq. (6) satisfies all the axioms, and thus is a persistence measure.

**Proof.** We show that $P(x;[t_s,t_e])$ is a persistence measure by proving that it satisfies Axioms A1-A4. Since $\alpha, \beta, \gamma \in (0, \infty)$ do not affect the results, we omit them from the proofs for readability.

- **Axiom A1.** If $|O_x| = 0$, then $P(x;[t_s,t_e]) = 0$. If $|O_x| > 0$, then $P(x;[t_s,t_e]) > 0$, by Lemma 1. Therefore, since $|O_x| > 0$, if $P(x;[t_s,t_e]) = 0$, then $|O_x| = 0$.
- **Axiom A2.** Since $|\{t_f,t_j\}| \leq |\{t_s,t_e\}|$, $W(x;[t_s,t_e]) \leq 1$. Also, $H(\Gamma_x) \leq \log(|O_x|)$ [8], so $S(x;[t_s,t_e]) \leq 2$. Thus, these bounded terms do not affect the limit, and $\lim_{|O_x| \to \infty} \log_{10} |O_x| + 1 = \infty$. Thus, unaffected by $\alpha, \beta, \gamma \in (0, \infty)$, $\lim_{|O_x| \to \infty} P(x;[t_s,t_e]) = \infty$.
- **Axiom A3.** First, $W(t_s',t_e') = \frac{t_s' - t_f + t_j - c}{t_s' - t_f + t_j - c - t_s' - t_j} = W(t_s,t_e)$. Also, if $|O_x| = |O_x|$, then $S(x';[t_s',t_e']) = S(x;[t_s,t_e])$. Each new gap $g_i' = t_i + c - t_j = g_i + 3.1$, so shifting the occurrences does not change their spread: $S(x';[t_s',t_e']) = S(x;[t_s,t_e])$. As a result, the persistence remains the same.
- **Axiom A4.** By definition, $t_s \leq t_s' \leq t_f$ and $t_e' \geq t_e' \geq t_e$. This implies that $W(x;[t_s,t_e]) \leq W(x;[t_s',t_e']) $ $\leq W(x;[t_f,[t_s,t_e])$, with equality iff $t_f = t_i$ and $t_e = t_e$. Since all occurrences fall in $[t_f,[t_s,t_e)$, by definition, the frequency and spread terms remain the same upon shrinking the measurement interval. Therefore, $P(x;[t_s,t_e]) \leq P(x;[t_s',t_e']) \leq P(x;[t_f,[t_s,t_e])$, with equality iff $t_f = t_i$ and $t_e = t_e$.

**Theorem 2.** Our persistence measure $P(x;[t_s,t_e])$, Eq. (6), satisfies the three desired properties P1-P3.

**Proof.** We prove each property separately:

- **Property P1.** First, since $|O_x^n| = |O_x| = n$, $F(x^n;[t_s,t_e]) = F(x^n;[t_s,t_e])$. Also, for $|\Gamma_x^n| = |\Gamma_x| = n-1$ uniform gaps, $H(\Gamma_x^n) = H(\Gamma_x) = \log(n-1)$. Thus, $S(x^n;[t_s,t_e]) = S(x^n;[t_s,t_e])$. Since $|\{t_f,t_j\}| < |\{t_f,t_j\}|$ and $|\{t_s,t_e\}|$ is fixed, $W(x^n;[t_s,t_e]) = \log_2(n)$, Therefore, $P(x^n;[t_s,t_e]) = P(x;[t_s,t_e])$.

- **Property P2.** We prove this by case. For $n = 0$, $P(x;[t_s,t_e]) = 0$ by Axiom A1. For $n \geq 1$, $W(x^n;[t_s,t_e]) = W(x^n;[t_s,t_e])$, so $\{t_f,t_j\} = \{t_f,t_j\}$, and $|\Gamma_x| = n-1$. When $n = 1$, then $S(x;[t_s,t_e]) = 1 \leq S(x^{n+1};[t_s,t_e]) = 2 \leq F(x^n;[t_s,t_e]) = 2 \log_2(2) = 2$. For $n > 1$, because of uniform gaps, $H(\Gamma_x^n) = H(\Gamma_x) = \log(n)$, Therefore, $S(x^n;[t_s,t_e]) = S(x^n;[t_s,t_e]) = 1$ for $n \geq 1$. Lastly, $F(x^n;[t_s,t_e]) < F(x^n;[t_s,t_e]) = 0$ by combining the three terms, we prove the claim.

- **Property P3.** Since $|O_x^n| = |O_x|$, $F(x^n;[t_s,t_e]) = F(x^n;[t_s,t_e])$. Also, $|\Gamma_x^n| = |\Gamma_x^n|$, so $W(x^n;[t_s,t_e]) = W(x^n;[t_s,t_e])$. Moreover, $H(\Gamma_x^n) \leq H(\Gamma_x^n)$, with equality iff the gaps $g_i \in \Gamma_x^n$'s are uniform. Thus, $S(x^n;[t_s,t_e]) = S(x^n;[t_s,t_e])$, with equality iff the gaps $g_i \in \Gamma_x^n$'s are uniform, in which case $\Gamma_x = \Gamma_x^n$.


Persistence Heuristic in Data Streams [9]. A simple persistence heuristic was proposed in [9]: an item in a data stream is considered persistent if it occurs at least once in a large number of predefined, equally-sized observation periods (intervals). We show that this heuristic violates axioms A2-A4 via counter-examples. As the number of unique occurrences tends to infinity, the heuristic tends to the number of intervals, not infinity (A2 violation). If an item has two occurrences in the same interval, they can be shifted such that one falls in a new period, in which case the heuristic grows (A3 violation). Finally, for an item with two occurrences in different intervals, after shrinking \([t_s, t_e]\) towards \([t_f, t_l]_x\) and re-dividing into the predefined number of intervals, they could still fall in different intervals. Thus persistence would not increase (A4 violation).

4 PROPOSED METHOD: PENMINER

With our proposed persistence measure in Eq. (6), and the definitions in § 3.1, we can now define the offline and streaming (online) versions of our problem precisely.

Problem 2 (Offline/Streaming Persistent Activity Mining). Given an edge stream, \(\mathcal{S}\), a maximum duration \(\delta_{\text{max}}\) and a maximum snippet size \(k_{\text{max}}\),

- [Offline] output the persistence \(P(x; \text{START}(), \text{END}())\)
- [Streaming] maintain the persistence \(P(x; \text{START}(), t)\) of every \((\delta_{\text{max}}, k_{\text{max}})\)-activity snippet \(x\) observed in the whole stream.

We next introduce our offline and streaming algorithms, oPENminer and sPENminer, to solve this problem. We first discuss extracting all \((\delta_{\text{max}}, k_{\text{max}})\)-snippets from a stream \(\mathcal{S}\), followed by the details of each variant. We give the main steps for PENminer, which takes as a parameter which variant to use, in Algorithm 1, and detailed pseudocode for reproducibility in § A.2.

Activity Snippet Extraction (line 2). Each time a new edge-update \(u_{\text{new}}\) arrives (at time \(t\)), the procedure ExtractNewSnippets is called. The procedure maintains the window \(W\), which consists of all updates \(g\) that occurred within the last \(\delta_{\text{max}}\) time units, and removes all others. It then adds the new update \(u_{\text{new}}\) (which is 0-time-units in the past). In addition to maintaining the window, the procedure extracts all valid snippets from the window. A valid snippet must be connected, have duration \(\delta \leq \delta_{\text{max}}\), and size \(k \leq k_{\text{max}}\). Since all stale updates have been removed from \(W\), \(\delta_{\text{max}}\) is already enforced. Any new snippet instance must contain \(u_{\text{new}}\). The singleton snippet containing just \(u_{\text{new}}\) is created, and then snippets of size \(k = 2, \ldots, k_{\text{max}}\) are constructed smallest to largest, such that the nodes in each snippet are connected.

Offline Algorithm (lines 4 and 8). For the offline version of Problem 2, oPENminer maintains the set of occurrences of each activity snippet extracted from the stream. Then, when the end of the stream is reached, it computes and outputs the persistence of each snippet \(x\) in \([\text{START}(), \text{END}()]\) with Eq. (6) as \(P(x; \text{START}(), \text{END}())\).

Streaming Algorithm (line 6). For the online version of Problem 2, there are two cases to handle: (1) update the persistence of snippet \(x\) when it occurs at time \(t\), and (2) return the correct persistence of a snippet \(x\) if it is queried at any other time \(t\). We maintain in memory a constant amount of information on each snippet: the total number of its occurrences, \(|O_x|\), the number of gaps between occurrences, \(|G_x|\), the time of its first and last occurrences, \(t_f, t_l\), and its persistence when it last occurred, \(P(x; \text{START}(), t_l)\).

- Case 1. Updating Persistence upon Occurrence. Since there is a new occurrence at the current time \(t\) and we maintain the first occurrence of each snippet, we can compute the width function \(W(x; [t_s, t_e]) = \frac{|\{t_r, t_l+1\}|}{|\text{START}(), t_l+1|} \) from Eq. (2). Since we maintain the number of occurrences, \(|O_x|\), we can obtain the frequency \(F(x; \text{START}(), t) = \log_\alpha |O_x| + 1 \) from Eq. (3). By maintaining the number of gaps between occurrences, we know whether \(|G_x| \in [0, 1)\) (in Eq. (5)) and can compute \(\log |G_x|\). In order to compute \(H(F)\) in Eq. (4)—we show how the entropy of the distribution induced by a snippet \(x\)'s gap widths can be computed incrementally, as new occurrences create new gaps. Specifically, the entropy of a new gap is computed from (1) the previous entropy, (2) the previous normalizing constant \(Z\), and (3) the new gap \(g_{n+1}\). This is stated in Thm. 3 and proved via the following two Lemmas.

Let \(p_i = \frac{\gamma_i}{Z}\), where \(Z = \sum_{i=1}^{n} g_i\), be a probability mass function on a set of \(n\) gaps, \(G_x\), induced by normalizing each gap by \(Z\).

**Lemma 2.** The entropy \(H(p) = \log Z - \frac{1}{Z} \sum_{i=1}^{n} g_i \log g_i\).

**Lemma 3.** For the new set of gaps \(G' = G_x \cup \{g_{n+1}\}\), and normalizing constant \(Z' = Z + g_{n+1}\), the corresponding pmf \(p'\) has entropy \(H(p') = \frac{Z'}{Z} \log Z' - \frac{g_{n+1}}{Z'} \log g_{n+1} + \left(\frac{1}{Z} - \frac{1}{Z'}\right) \log (Z - H(p))Z\), where \(H(p)\) is the entropy of \(p\), \(Z\) is the normalizing constant for \(p\), and \(g_{n+1}\) is the new gap.

**Theorem 3.** \(H(p') = H(p) + \frac{Z'}{Z} \log Z' - \frac{g_{n+1}}{Z'} \log g_{n+1} + \left(\frac{Z}{Z'} - \frac{Z}{Z} - 1\right) \log (Z - H(p))Z\), where \(H(p)\) is the entropy of \(p\), \(Z\) is the normalizing constant for \(p\), and \(g_{n+1}\) is the new gap.

**Proof.** Let \(\Delta = H(p') - H(p)\) be the change in entropy and \(X = \sum_{i=1}^{n} g_i = \log Z\). By Lemmas 2-3, we obtain \(\Delta = \frac{Z'}{Z} \log Z' - \log Z - \frac{g_{n+1}}{Z'} \log g_{n+1} + \left(\frac{Z}{Z'} - \frac{Z}{Z} - 1\right) \log (Z - H(p))Z\). Plugging, this into \(H(p') = H(p) + \Delta\) completes the proof.

The new gap \(g_{n+1}\) is the time from the previous occurrence of \(x\) (which we know since we maintain \(t_l\)) to the current time \(t\). The normalizing constant \(Z\) is the time from the first occurrence \(t_f\) to the previous occurrence \(t_l\), both of which we maintain.

**Case 2. Querying Persistence Without a New Occurrence.** Without a new occurrence, the frequency \(F(x; \text{START}(), t)\) and spread \(S(x; \text{START}(), t)\) do not change. From Eq. (2), we compute the width function, \(W(x; \text{START}(), t) = \frac{1}{Z}\), by replacing \(|\text{START}(), t_l|\) with \(|\text{START}(), t|\) in the denominator.

**Algorithm 1 PENminer ([\(\delta_{\text{max}}, k_{\text{max}}, \phi, \alpha, \beta, \gamma, \text{VARIANT}\)]\)

**Input:** Stream \(\mathcal{S}\), max snippet duration \(\delta_{\text{max}}\) and size \(k_{\text{max}}\), view \(\phi\), persistence exponents \(\alpha, \beta, \gamma\), the variant (oPENminer/sPENminer).

1. while \(u_{\text{new}} \in \mathcal{S}\) do
   2. if variant is oPENminer then
   3. for \(x \in \text{ExtractNewSnippets}(\mathcal{S}, u_{\text{new}}, t, \phi)\) do
   4. if variant is oPENminer then
   5. else
   6. Update \(P(x; \text{START}(), t_l)\) incrementally.
   7. Compute \(P(x; \text{START}(), \text{END}())\) for each \(x\) observed.
5 EVALUATION

We investigate the following research questions across multiple real networks (§ 5.1):

- **RQ1** What does the relationship between frequency and persistence reveal about activity and networks?
- **RQ2** Can sPENminer find anomalies in real-time?
- **RQ3** Can sPENminer process updates to a network at least as quickly as they arrive, and how does oPENminer scale with the number of edge updates in the stream?

In § 5.2 we investigate a core value of measuring persistence, which is the relationship between frequency and persistence. Specifically, we find that activity snippets with high persistence, but low frequency tend to correspond to subtle yet regular activity, which would be missed by measuring frequency alone. On the other hand, snippets with high frequency, but low persistence tend to be bursty. In § 5.3, we show how we can make analogous insights in real-time with sPENminer, and accurately find both subtle and bursty anomalies right when they occur. Finally, in § 5.4, we evaluate the maximum duration $\delta_{\text{max}}$ and maximum size $k_{\text{max}}$ parameters, and demonstrate that sPENminer processes edges in each stream 10K to 360K times faster than the rate of that stream, and oPENminer scales linearly with the number of edge-updates processed.

5.1 Data

We utilize a diverse set of evolving networks (Tab. 2), including communication, transportation, computer, and social networks.

- The communication network Eu Email has timestamped edges denoting emails sent within a European research institute [18].
- Columbus Bike, Chicago Bike, and Boston Bike are networks encoding bike trips made in the bike-share systems in those three cities [1]. Each node is a bike station. When a bike leaves one station for another, an edge is inserted into the network; when the bike arrives at its destination, it is deleted. Thus, the networks capture en route bike-trips.
- Similarly, NYC Taxi [2] captures en route taxi trips in New York City, but in this case, nodes are city zones rather than stations.
- Edges in the social network Reddit correspond to timestamped references between subreddits (topical discussion boards) [18].
- DARPA IP [20] is an IP-IP network, where both normal network traffic and malicious attack traffic is present. Edges denote interactions between computers in the network.
- Stackoverflow [18] consists of interactions among users on the website Stackoverflow. The interactions are between users, and can be answers to questions, comments on questions, or comments on answers.

Table 2: Description of edge streams: number of edge-updates, number of nodes, whether it has edge deletions, number of unique edges, number of edge types, and average rate of the stream in updates/sec.

| Network   | V | E | Del | $|\ E| | rate (g/sec) |
|-----------|---|---|-----|---|---|----------------|
| Eu Email  | 322,332 | 986 | N | 24,929 | 1 | 0.04          |
| Columbus Bike | 534,998 | 74 | Y | 2,971 | 1 | 0.02          |
| Reddit    | 858,488 | 67,180 | N | 339,643 | 1 | 0.02          |
| Darpa IP  | 4,554,344 | 25,525 | N | 68,910 | 1 | 0.05          |
| Boston Bike | 17,421,182 | 476 | Y | 81,508 | 1 | 0.05          |
| Chicago Bike | 33,331,104 | 712 | N | 171,651 | 1 | 0.25          |
| Stackoverflow | 63,497,050 | 2,601,977 | N | 36,233,430 | 3 | 0.27          |
| NYC Taxi  | 3,077,990,404 | 265 | Y | 60,750 | 1 | 9.29          |

Figure 4: Boston (a) and Columbus (b) bike networks. Representative timelines from various parts of the PvF plots demonstrate how persistent, bursty, and subtle activity can be identified.

5.2 RQ1: Persistent vs. Frequent

The relationship between frequency and persistence allows for a more complete view of activity in networks. We show this via a Persistence vs. Frequency, or PvF, plot of activity snippets, with snippet frequency, $|Q_\gamma|$, in log-scale on the y-axis, and persistence $P(x;\alpha,\delta_{\text{start}},\delta_{\text{end}})$ on the x-axis. Activity snippets with unusually high persistence relative to their frequency fall towards the lower-right of the plot. Those with unusually high frequency relative to their persistence fall towards the upper-left. We call the former type of activity snippet subtly persistent, and the latter bursty (whether a single burst or periodic bursts). We analyze persistent, subtly persistent, and bursty activity snippets in several networks via PvF plots, and show what they correspond to in each context. To see what the baseline of frequency would give, the variation of points along the y-axis only can be studied.

5.2.1 Transportation Networks. Setup. We generate PvF plots for bike trips in Boston, MA, Columbus, OH, and Taxi trips in New York City (NYC) for the two weeks surrounding Hurricane Sandy (10/22/12-11/05/12). We use view $\phi = \text{ID}$. Since once a trip is complete the rider leaves the bike/taxi, sequences of edges are not linked. Thus, we set $k_{\text{max}} = 1$ (and $\delta_{\text{max}} = 1$), so that each activity snippet corresponds to a single bike/taxi trip. Next, in social networks, we use $k_{\text{max}} > 1$. We set $\alpha, \beta$ and $\gamma$ for visual clarity, and discuss how in the supplement on reproducibility (§ A.3).

Results. Results for Boston Bike and Columbus Bike are in Fig. 4. The most persistent bike trips (black) are both frequent and persistent. A representative timeline from each, where each tick is an occurrence of the snippet, is shown in the lower-right of each plot. In Boston, the most persistent trip is from Massachusetts Ave. in front of MIT, up the street to outside the Central subway station. This is a reasonable route for commuters to bike from MIT to the subway. In both networks, bursts (orange) reveal new bike stations opening. The new station in Boston opened right in front of MIT and became immediately popular, which led to a large burst of activity. With the station now established, its persistence should grow over time. For comparison, we show a representative point from the middle of the Columbus PuF plot, and a trip in Boston that is more persistent than expected given its frequency (blue).

In NYC Taxi, the bursty and subtly persistent snippets (Taxi trips) reveal which neighborhoods in NYC were most affected by Hurricane Sandy (Fig. 5). The hurricane made landfall in NYC around 8pm EST on 10/29 (shown with a dashed line). The bursty anomalies (orange) all correspond to neighborhoods that were brought to a
were canceled en masse. This has applications in city planning, since commute routes can be revealed during severe flooding. The decrease in trips prior to the storm is likely due to businesses closing in preparation. Many of the subtly persistent trips (blue) correspond to neighborhoods that were resilient despite the storm. The first example is Lenox Hill and East Harlem. While these neighborhoods border the East River, they extend inland, allowing plenty of roads for taxis to continue service. Lenox Hill hospital was also the subject of a study of the clinical response to Hurricane Sandy [29]. Possibly, the continuity in taxi trips was due to visits to the hospital. Similarly, the Upper West Side extends into inner-Manhattan, and Washington Heights is a significant distance north of where the hurricane made landfall.

**Takeaways on Transportation.** We find that subtly persistent activity often reveals commutes, because of their high regularity. This has applications in city planning, since commute routes can be good options for introducing ride-share programs. We also find that bursty activity often reveals major changes in the real world, such as severe weather disturbances or new routes becoming possible.

**5.2.2 Social Networks. Setup.** We use the social network Stackoverflow, where users interact by answering technical questions, commenting on questions, and commenting on answers. We first analyze a 3-month interval. For this analysis, we use the view \( \phi = \text{ID} \), set \( k_{\text{max}} = 3 \), and \( \delta_{\text{max}} = 1 \text{hr} \). Secondly, we analyze the entire dataset using \( \phi = \text{Order} \), with \( k_{\text{max}} = 3 \) and \( \delta_{\text{max}} = 15 \text{min} \). In the latter scenario, the snippets we analyze are a superset of the 3-node, 3-edge temporal motifs studied in [24]. We plot, in orange, snippets involving users commenting on questions, in black, those involving answers commenting on answers. We first analyze snippets that are subtlety persistent (blue), corresponding to neighborhoods that were resilient despite the storm. The first example is Lenox Hill and East Harlem. While these neighborhoods border the East River, they extend inland, allowing plenty of roads for taxis to continue service. Lenox Hill hospital was also the subject of a study of the clinical response to Hurricane Sandy [29]. Possibly, the continuity in taxi trips was due to visits to the hospital. Similarly, the Upper West Side extends into inner-Manhattan, and Washington Heights is a significant distance north of where the hurricane made landfall.

**Takeaways on Social Networks.** We find that subtly persistent activity often reveals commutes, because of their high regularity. This has applications in city planning, since commute routes can be good options for introducing ride-share programs. We also find that bursty activity often reveals major changes in the real world, such as severe weather disturbances or new routes becoming possible.

**5.2.2 Social Networks. Setup.** We use the social network Stackoverflow, where users interact by answering technical questions, commenting on questions, and commenting on answers. We first analyze a 3-month interval. For this analysis, we use the view \( \phi = \text{ID} \), set \( k_{\text{max}} = 3 \), and \( \delta_{\text{max}} = 1 \text{hr} \). Secondly, we analyze the entire dataset using \( \phi = \text{Order} \), with \( k_{\text{max}} = 3 \) and \( \delta_{\text{max}} = 15 \text{min} \). In the latter scenario, the snippets we analyze are a superset of the 3-node, 3-edge temporal motifs studied in [24]. We plot, in orange, snippets involving users commenting on questions, in black, those involving answers commenting on answers, and in blue, answering questions.

**Results.** Results are shown in Fig. 6. The timelines in (a) are for activity snippets among specific users. The bursty anomaly (orange) reveals users u72603 and u82199 (anonymized ids) commenting on u82199’s answer 36 times over the course of 1hr in the 3-month interval, then never interacting again. The timeline shown is zoomed-in on for clarity. The most persistent snippets (one shown in black) are users commenting on their own answers, suggesting that it is unusual for the same two distinct users to interact persistently over time. On the other hand, the subtly persistent anomaly (blue) reveals u1950 regularly answering u55747’s questions. We show the persistent, bursty, and subtly anomalous snippets in (i)–(iii).

We give the PVF plot used to identify these snippets in § A.4. The right plot uses the view \( \phi = \text{Order} \), where any users can form an occurrence of the activity snippet. Remarkably, we find that activity involving the three interaction types (commenting on questions, commenting on answers, and answering questions) fall in distinct places on the plot. Activity involving comments on questions tends to fall in the bursty region of the plot, while answers are the most subtly persistent. To better understand the phenomenon, we zoom in on the region shown in the upper-left. We observe that most activity snippets in this box target a single user (i.e., all discussion is directed towards that user), and involve at least two distinct users. We give examples in (iv)–(v) of Fig. 6. In (iv), a user comments on another’s post, the second user responds, and the original user comments again. In (v), the second user responds twice to the original comment. These snippets capture natural, technical discussions among users. Remarkably, these discussions center around questions (orange) and answers (black) at roughly equal frequencies, but they occur with higher persistence for answers. We conjecture that this is because, when a question is asked, it triggers a flurry of comments on it; but once the question is answered, these tend to die out. On the other hand, comments on answers persist, because new users may have the same question, and have follow-up questions, even months after the question has been answered. Since both have indistinguishable frequencies (10^3–10^5), this subtle difference in behavior is not revealed by frequency alone.

**Takeaways on Social Networks.** We find that subtly persistent activity can reveal users who do not interact often, but do regularly—i.e., similar users who are missed when only looking at how many times they interact. We also find that some types of posts promote continual activity, while others trigger bursts.

**5.3 RQ2: Anomalies in Real Time**

We next demonstrate that bursty and subtly persistent snippets can be identified in real time, right when they occur.

**5.3.1 Generating Anomaly Scores.** To identify anomalies automatically in real-time (without visually inspecting PVF plots as in § 5.2), we use the following process. When an activity snippet x appears in the stream at time t, we generate a 2D point <frequency,
We describe the exact injection procedure in § A.6. Each of the
synthetic bike trips that simulate infrequent, but lasting and sur-
prisingly regular traffic, analogous to the first taxi trip in Fig. 7.

We use sPENminer with $k_{\text{max}} = 1$, and set ($\alpha$, $\beta$, $\gamma$) values, which we give in § A.8. At each time $t$, we compute the anomaly scores of each ground-truth snippet (§ 5.3.1), and compare the scores to the median and standard deviation of the anomaly scores of all points seen to that point. We label an activity snippet at time $t$ as a level 1, 2, or 3 anomaly if it is 1 to 2, 2 to 3, or 3+ standard deviations above the median score respectively.

Results. The results in Fig. 7 for Reddit (left) and NYC Taxi (right) show level 1, 2, and 3 anomalies colored green, orange, and maroon.

The subtly persistent snippet in Reddit corresponds to the subreddit $r/electronic_cigarette$ referencing $r/ecrpoker$. This is the 3rd most persistent snippet overall, but only the 252nd most frequent. References between these subreddits occur remarkably regularly. Upon investigation, we found that $r/ecrpoker$ was formed by a popular user in $r/electronic_cigarette$. We conjecture that the snippet corresponds to this user, or their followers, promoting the content of the other subreddit. The bursty anomaly corresponds to $r/nightly_pick$ referencing $r/hockey$, presumably picking winners for each night’s hockey game. The bursts align with hockey seasons. sPENminer consistently identifies this activity snippet as anomalous. Furthermore, the anomaly score decreases as expected over time, since as the bursts return yearly during hockey season, the snippet becomes more persistent. The third snippet is neither bursty nor regular, and it is correctly not flagged as an anomaly.

In NYC, the subtly persistent anomaly reveals a taxi trip from Kew Gardens Hills in Queens, to Manhattan, near the United Nations building. The taxi trip is repeated every day shortly after midnight, and is almost never taken at any other time. The nature of the trip is unknown, but surprisingly regular. The score, as expected, grows over time, as the continued regularity increases anomalousness. The bursty anomaly captures taxi trips departing from and arriving at the zone containing the NYC Taxi & Limousine Commission. The Taxi Commission inspects taxis and is open 5 days a week, which suggests that the bursts correspond to test-drives of taxis for inspection during business hours. sPENminer consistently identifies these bursts. Again, the third snippet is neither bursty nor regular, and is not often flagged as an anomaly.

5.3.3 Real-time Quantitative Anomaly Detection. Setup. We quantitatively analyze sPENminer’s performance at identifying both subtle and bursty anomalies. For subtle anomalies, we use three months of Chicago Bike, from 01/2014 to 03/2014, and inject 50 synthetic bike trips that simulate infrequent, but lasting and surprisingly regular traffic, analogous to the first taxi trip in Fig. 7. We describe the exact injection procedure in § A.6. Each of the 50 anomalous trips occurs repeatedly, and the task is to identify the occurrences of all 50 bike trips. We average results over ten random injection sets. For bursty anomalies, we use the DARPA IP network commonly used for the task [7, 11]. In this dataset, 2.7 million edges correspond to various bursty network attacks (e.g., denial of service). The goal is to identify edges that are part of these attacks. We use the same $\alpha$, $\beta$, and $\gamma$ as § 5.3.2, since we found them useful for visually identifying bursty and subtly persistent snippets.

Baselines. (1) $\text{Freq}$ scores snippets as their number of occurrences, divided by the total number of all snippet occurrences. It can be thought of as [24] with motifs extended to activity snippets. (2) $\text{SedanSpot}$ [11] and (3) $\text{Midas-R}$ [7], state-of-the-art methods for bursty anomalies in edge-streams, output an anomaly score for each edge update. We set parameters as in the respective papers, and use the authors’ code. (4) $\text{DS}$ adapts the heuristic persistence of an item in a data-stream [9]. We apply $\text{DS}$ exactly as PENminer, but replace the persistence in the 2D point with their heuristic (described in § 3.4). For consistency, we follow the authors’ suggestion of dividing the stream into 60 measurement periods, even though this unrealistically assumes that the stream length is known a priori.

Results. We give results in Tab. 3. sPENminer outperforms baselines at finding subtly persistent anomalies. Simultaneously, for bursty anomalies, it performs competitively with $\text{Midas-R}$, which is designed specifically for the task of finding bursty anomalies. On the other hand, the methods targeting bursts do not perform competitively at finding subtle anomalies. All but one result are statistically significant at a 0.01 $p$-value in a paired $t$-test.
We evaluate whether sPENminer can process edges at least as quickly as they arrive in a stream, while allowing snippets to be reasonably sized ($\delta_{\text{max}}$) and take a reasonable amount of time to form ($\delta_{\text{max}}$). We then analyze how oPENminer scales with the number of edge-updates. We discuss hardware in § A.7.

### 5.4 RQ3: Efficiency and Scalability

We evaluate whether sPENminer can process edges at least as quickly as they arrive in a stream, while allowing snippets to be reasonably sized ($\delta_{\text{max}}$) and take a reasonable amount of time to form ($\delta_{\text{max}}$). We then analyze how oPENminer scales with the number of edge-updates. We discuss hardware in § A.7.

#### 5.4.1 sPENminer Efficiency. Setup. To evaluate sPENminer’s efficiency over different $\delta_{\text{max}}$ and $k_{\text{max}}$ settings, we create plots for $k_{\text{max}} = \{1, 2, 3\}$. In each, we fix $k_{\text{max}}$ and vary $\delta_{\text{max}} = \{60, 120, 180, 300, 600, 900, 1800\}$. We show edges processed per second for each parameter, for all datasets with rates less than 1 update/sec (since streams with significantly different rates are not comparable). Each point is averaged over 5 random intervals of 100K edge-updates (the same intervals across parameters).

**Results.** We show the results in Fig. 8. For $k_{\text{max}} = 1$, activity snippets have duration $\delta = 0$, which is why in the first plot, the edges processed per second is consistent over all $\delta_{\text{max}}$. Across all streams, and parameters, sPENminer processes edge-updates 10K to 360K times faster than the rate of the corresponding stream.

#### 5.4.2 oPENminer Scalability. Setup. We evaluate how oPENminer scales with increasingly more edge-updates in Stackoverflow, our network with the most edges and nodes. We process the first 100K, 500K, 1M, 10M, 25M, and 50M edge-updates 5 times, and report the average runtime in seconds (Fig. 9). We fix $\delta_{\text{max}} = 600$ sec (10 min) and evaluate $k_{\text{max}} \in \{1, 2, 3\}$.

**Result.** oPENminer scales linearly as the network grows.

### 6 CONCLUSION

In this paper, we propose mining persistent activity in continually evolving networks. Our precise, theoretical definition of persistence captures, beyond the aggregate number of occurrences, for how long and how regularly the activity has occurred. We propose PENminer (both offline and streaming variants) to measure the persistence of activity in evolving networks, and use it to gain a better understanding of networks by revealing activity that frequency alone could not, from infrequent but surprisingly regular trips in traffic networks to heated conversations in social networks. Future work includes further developing persistence-based anomaly detection, and techniques for automatic parameter tuning ($\alpha$, $\beta$, $\gamma$).

### ACKNOWLEDGEMENTS

This work is supported by an NSF GRFP Fellowship, the NSF under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, and Adobe, Amazon, and Google faculty awards.

### REFERENCES

A SUPPLEMENT ON REPRODUCIBILITY

A.1 Complexity Analysis

In the special case of $k_{\text{max}} = 1$, the window $\mathcal{W}$ need not be maintained (only singleton snippets are extracted). Thus, the per-update and total complexity (of oPENminer) are $O(1)$ and $O(|\mathcal{S}|)$ respectively. We now discuss $k_{\text{max}} > 1$. To process a new update $u_{\text{new}}$, the only non-constant cost comes from extracting new snippets in line 4 (lines 5-8 are $O(1)$). Let $\mu$ be the average rate of the stream $\mathcal{S}$ in updates per second. Then the average number of updates in a window $\mathcal{W}$ of width $w = \delta_{\text{max}}$ seconds is equal to $\mu \cdot \delta_{\text{max}}$. Lines 14-16 are $O(\mu \cdot \delta_{\text{max}})$. While new snippets must be connected (cf. §4), in the worst case all previous $\mu \cdot \delta_{\text{max}}$ updates are connected with $u_{\text{new}}$. Thus, there are $O\left(\sum_{k=1}^{k_{\text{max}}} \binom{\mu \cdot \delta_{\text{max}}}{k-1}\right)$ new snippets to extract (lines 18-20), which dominates $O(\mu \cdot \delta_{\text{max}})$. Consequently, the per-update time can be controlled by choosing $\delta_{\text{max}}$ to be reasonable based on the stream’s rate. For oPENminer, the total time complexity to process $|\mathcal{S}|$ updates is $O\left(|\mathcal{S}|(\mu \cdot \delta_{\text{max}})^{-1}\right)$.

A.2 Detailed Pseudocode

Algorithm 2 PENminer ($\mathcal{S}$, $\delta_{\text{max}}$, $k_{\text{max}}$, $\phi$, $\alpha$, $\beta$, $\gamma$, VARIANT)

Input: Stream $\mathcal{S}$, max snippet duration $\delta_{\text{max}}$ and size $k_{\text{max}}$, view $\phi$, persistence exponents $\alpha$, $\beta$, $\gamma$, the variant (oPENminer/sPENminer).

1: $\mathcal{W} \leftarrow \delta_{\text{max}}$ $\triangleright$ The window size enforces the maximum duration
2: $\mathcal{W} \leftarrow \emptyset$ $\triangleright$ Window is empty initially
3: while $u_{\text{new}} \in \mathcal{W}$ do $\triangleright$ While there is a new update in the stream
4: for $x \in \text{ExtractNewSnippets($\mathcal{W}$, $u_{\text{new}}$, $t$, $\phi$)}$ do
5: if variant is oPENminer then
6: $\mathcal{O}_x \leftarrow \mathcal{O}_x \cup \{t\}$ $\triangleright$ Add the new occurrence’s timestamp
7: else
8: Update $P(x; \text{start}(\mathcal{S}), t)$ via Thm. 3
9: if variant is oPENminer then
10: for each $x$ do
11: Compute $P(x; \text{start}(\mathcal{S}), \text{end}(\mathcal{S}))$
12: procedure ExtractNewSnippets($\mathcal{W}$, $u_{\text{new}}$, $t$, $\phi$)
13: $\mathcal{X}_{\text{extracted}} \leftarrow \{\{u_{\text{new}}\}\}$ $\triangleright$ Add the singleton snippet for $u_{\text{new}}$
14: for $u \in \mathcal{W}$ do
15: if $t > \text{timestamp}(u)$ then
16: $\mathcal{W} \leftarrow \mathcal{W} \cup \{u\}$ $\triangleright$ Remove stale updates
17: $\mathcal{W} \leftarrow \mathcal{W} \cup \{u_{\text{new}}\}$ $\triangleright$ Add the new update
18: for $k = 2, \ldots, k_{\text{max}}$ do
19: for each new size $k$ snippet $x$ do
20: $\mathcal{X}_{\text{extracted}} \leftarrow \mathcal{X}_{\text{extracted}} \cup \{x\}$
21: return $\mathcal{X}_{\text{extracted}}$

A.3 Choosing $\alpha$, $\beta$, $\gamma$

This section provides analysis of the exponents $\alpha$, $\beta$, and $\gamma$. At the end of the section, we give suggestions for practitioners.

A.3.1 Rank Correlation Between Persistence and Components. We first show how varying the exponents affects how much each component contributes to persistence. To do so, we compare the ranking of snippets in descending order on $W(\cdot)$, $F(\cdot)$, and $S(\cdot)$, with the ranking in descending order based on persistence $P(\cdot)$. We use Kendall-Tau rank-correlation to compare rankings. For each component of persistence $W(\cdot)$, $F(\cdot)$, and $S(\cdot)$, we vary its corresponding exponent $\alpha$, $\beta$, or $\gamma$ over the values $\{0.2, 0.5, 1, 2.0, 5.0\}$, while fixing the other two at 1.

Figure 11: Kendall Tau rank correlation between snippets ranked by components of persistence and persistence itself, over various values of exponents. In general, as one exponent is increased, and the others fixed, the corresponding component becomes more correlated with persistence.
Table 4: Additional results at identifying subtly persistent and bursty anomalies, showing the effect of parameters \((\alpha, \beta, \gamma)\).

<table>
<thead>
<tr>
<th>Metric</th>
<th>(1, 1, 1)</th>
<th>(0.2, 1, 1)</th>
<th>(1, 0.2, 1)</th>
<th>(1, 1, 0.2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUC</td>
<td>0.709 ± 0.02</td>
<td>0.801 ± 0.02</td>
<td>0.712 ± 0.02</td>
<td>0.686 ± 0.03</td>
</tr>
<tr>
<td>F1@100</td>
<td>0.006 ± 0.00</td>
<td>0.009 ± 0.00</td>
<td>0.005 ± 0.00</td>
<td>0.007 ± 0.00</td>
</tr>
<tr>
<td>F1@1K</td>
<td>0.028 ± 0.00</td>
<td>0.051 ± 0.00</td>
<td>0.029 ± 0.00</td>
<td>0.028 ± 0.01</td>
</tr>
<tr>
<td>F1@2K</td>
<td>0.042 ± 0.01</td>
<td>0.076 ± 0.01</td>
<td>0.043 ± 0.01</td>
<td>0.041 ± 0.01</td>
</tr>
<tr>
<td>Bursty</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUC</td>
<td>0.856 ± 0.01</td>
<td>0.867 ± 0.01</td>
<td>0.831 ± 0.01</td>
<td>0.853 ± 0.01</td>
</tr>
<tr>
<td>F1@500K</td>
<td>0.307 ± 0.00</td>
<td>0.307 ± 0.00</td>
<td>0.307 ± 0.00</td>
<td>0.307 ± 0.00</td>
</tr>
<tr>
<td>F1@1M</td>
<td>0.525 ± 0.00</td>
<td>0.527 ± 0.00</td>
<td>0.516 ± 0.01</td>
<td>0.524 ± 0.00</td>
</tr>
<tr>
<td>F1@2M</td>
<td>0.763 ± 0.01</td>
<td>0.783 ± 0.01</td>
<td>0.742 ± 0.02</td>
<td>0.756 ± 0.01</td>
</tr>
</tbody>
</table>

Table 5: Reference of parameters used in our code for figures and tables reported in §5. For maximum size of \(k_{\text{max}} = 1\), the duration of a snippet is always 0, in which case the maximum duration can be set arbitrarily without affecting results.

<table>
<thead>
<tr>
<th>Tab./Fig.</th>
<th>(k_{\text{max}})</th>
<th>(\delta_{\text{max}})</th>
<th>Variant</th>
<th>View</th>
<th>(\phi)</th>
<th>(\alpha)</th>
<th>(\beta)</th>
<th>(\gamma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 4 (Boston)</td>
<td>1</td>
<td>N/A</td>
<td>oPENminer</td>
<td>ID</td>
<td>1</td>
<td>0.5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Fig. 4 (Columbus)</td>
<td>1</td>
<td>N/A</td>
<td>oPENminer</td>
<td>ID</td>
<td>2</td>
<td>0.5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Fig. 5</td>
<td>1</td>
<td>N/A</td>
<td>oPENminer</td>
<td>ID</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Fig. 6 (left)</td>
<td>3</td>
<td>3600</td>
<td>oPENminer</td>
<td>ID</td>
<td>1</td>
<td>0.5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Fig. 6 (right)</td>
<td>3</td>
<td>900</td>
<td>oPENminer</td>
<td>ORDER</td>
<td>2</td>
<td>0.5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Fig. 7</td>
<td>1</td>
<td>N/A</td>
<td>sPENminer</td>
<td>ID</td>
<td>1</td>
<td>0.2</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>