Partitioning Communication Streams into Graph Snapshots

Jeremy D. Wendt, Richard V. Field, Jr., Cynthia A. Phillips, *Member, IEEE,* Arvind Prasadan, Tegan Wilson, Sucheta Soundarajan Sanjukta Bhowmick

***Abstract*—We present EASEE (Edge Advertisements into Snapshots using Evolving Expectations) for partitioning streaming communication data into static graph snapshots. Given streaming communication events (A talks to B), EASEE identifies when events suffice for a static graph (a *snapshot*). EASEE uses combinatorial statistical models to adaptively find when a snapshot is stable, while watching for significant data shifts – indicating a new snapshot should begin. If snapshots are not found carefully, they poorly represent the underlying data – and downstream graph analytics fail: We show a community detection example.**

We demonstrate EASEE's strengths against several real-world datasets, and its accuracy against known-answer synthetic datasets. Synthetic datasets' results show that (1) EASEE finds known-answer data shifts very quickly; and (2) ignoring these shifts drastically affects analytics on resulting snapshots. We show that previous work misses these shifts. Further, we evaluate EASEE against seven real-world datasets (330K to 2.5B events), and find snapshot-over-time behaviors missed by previous works. Finally, we show that the resulting snapshots' measured properties (e.g., graph density) are altered by how snapshots are identified from the communication event stream. In particular, EASEE's snapshots do not generally "densify" over time, contradicting previous influential results that used simpler partitioning methods.

***Index Terms*—datasets, graph sampling, social networks, network evolution**

I. Introduction

Many static graphs are built from underlying temporal data.Each temporal event (e.g., u talks to v at time t) can be viewed as an "edge advertisement" (EA). EA datasets can be quite massive – covering years of data – and are often converted into a series of static graphs (or *snapshots*). Although techniques exist to create snapshots from EAs [1]–[7], the many practitioners we consulted generally use either all their data, or an amount that has worked previously for their applications. Herein, we demonstrate a new, combinatorial statistics-based technique which forms snapshots from EA streams.

For successful network analysis, the input snapshots must be created to faithfully represent the underlying data – as shown by extensive research [8]–[12]. For instance (as we show in Sec. VI-B), community detection can find the correct community assignments only when the snapshots it receives match the underlying EAs stability and changes: Since the underlying EA streams generally change with time, the series of snapshots must reflect those changes. We introduce a statistical model for monitoring snapshot properties from the EA stream and use it to faithfully find snapshot boundaries.

When building static snapshots from streaming data, the principle decision is which EAs to include in a snapshot. In this work (and many previous), this means choosing the time interval over which to accumulate edges from EAs.

An additional challenge for EA datasets is that the underlying edges and nodes may be joining and leaving during a snapshot's interval. We focus on two competing interval size concerns. An interval that is too small (undersampling) can lead to a grossly incomplete graph; an interval that is too large (oversampling) can hide important changes or shifts in the data.

As an example of oversampling, consider computer network traffic in a wireless-enabled building. New computers enter and exit the network, and transfer between wireless access points (WAPs) as they travel through the building. With a too large interval, a static graph representation may have all WAPs connected through a single well-traveled computer even though the computer is never connected to all at once. Previous work identifies moments of catastrophic shift with massive edge or node changes (e.g., [1]–[7]), but does not identify frequent small changes. A key feature of our technique is its ability to assess and trade-off both kinds of error.

Contribution: We create representative graph snapshots from EA data by presenting *a broadly applicable, interpretable, concise model for building one or more static graphs from an EA stream.* Our proposed approach, Edge Advertisements into Snapshots using Evolving Expectations (or EASEE), has the following features that make it suitable for analyzing EA data:

1) *Adaptively Finding Snapshot Intervals* (Sec. III). EASEE adaptively finds appropriate snapshot intervals to minimize the effects of over- and undersampling. It selects a minimal *sufficient* interval when the current snapshot stabilizes. It merges neighboring intervals if they are similar enough.
2) *Predicting Future Graph Sizes* (Sec. III-A). EASEE predicts near-future growth in numbers of nodes and edges while building a snapshot. It can quickly test these predictions to measure how well the model fits the communication data (Sec. V-A).
3) *Executing in Real Time* (Sec. VI). EASEE runs on-line and in real time with incoming communication data with minimal compute overhead. This is important when analyzing quickly evolving data streams.
4) *Requiring Minimal Parameter Tuning* (Sec. V). Many adaptive techniques require considerable expertise to

J. Wendt is the corresponding author. J. Wendt, R.V. Field, C. Phillips and A. Prasadan: Sandia National Laboratories, Albuquerque, NM 87185. email:{jdwendt,rvfield,caphill,aprasad}@sandia.gov.

T. Wilson: Cornell University, Ithaca, NY 14853, teganwilson@cs.cornell.edu

S. Soundarajan: Syracuse University, Syracuse, NY 13244, susounda@syr.edu

S. Bhowmick: University of North Texas, Denton, TX 76203, sanjukta.bhowmick@unt.edu

properly set parameters. EASEE has only three parameters (recent history size, smoothing window size, and merging threshold). We show how to easily find suitable values for the first and third. We show that choosing a reasonable value for the second provides good results.

5) *Identifying Underlying Data Shifts* (Sec. VI). EASEE monitors underlying EA properties, and directly identifies times of drastic change, which cause EASEE to start a new snapshot. We demonstrate this with real-world examples, and quantify errors with synthetic datasets. We also show that previous works do not evince such sensitivity.
6) *Recognizing Problematic Datasets* (Sec. VI-E). Our focus on balancing under- and oversampling enables EASEE to recognize datasets that shift so rapidly that quality snapshots can't be generated. With such datasets, static snapshots likely create graphs with more paths that were never active together than those that were.

We demonstrate EASEE's accuracy against two known-answer synthetic datasets – comparing against some previous work. We also show its effectiveness against seven varying-size, real-world datasets – one with over 2 billion EAs.

Finally, our results add caveats to previous analyses of such streaming datasets. Prior work has argued that graphs from streaming data densify over time [13]. We repeat experiments from that previous work and add EASEE's output to the analysis (Sec. VII). We demonstrate that the previous statements about densification are not generally true on the datasets we analyze – both with EASEE and the previous fixed-width snapshot intervals. However, our results show times where continuing to develop a single static graph would blur two distinct behavior periods into a single static representation. Thus, while densification sometimes occurs, some of its reported occurrences may be due to merging wildly varying underlying graphs. See Sec. VIII for further discussion.

Organization: After reviewing related work in Sec. II, we describe the EASEE model in Sec. III. We present real-world datasets in Sec. IV, and use them to set model parameters in Sec. V. Sec. VI describes several experiments: (1) two known-answer synthetic datasets show that EASEE performs better than previous works; (2) real-world datasets (a) measure sensitivity to starting position in a datastream; (b) visualize our merging step; and (c) identify datasets that should *not* be converted into static snapshots. Sec. VII re-analyzes conclusions from previous work [13] in light of EASEE snapshots. In Sec. VIII, we discuss lessons from EASEE and conclude.

II. Related Work

Previous works have focused on three facets of temporal communication streams. Some analyzed how varying snapshot intervals affect the resulting graphs – without proposing techniques for setting the interval. Others avoid creating static snapshots and instead propose new techniques to learn directly from the temporal data: This field is called "Temporal Networks". Closest to this paper's work, some developed techniques to find graph snapshot intervals directly from temporal data. We describe each briefly herein.

Varying Snapshot Interval: Researchers have studied how graph properties evolve when they aggregate EAs into static snapshots. Most consider cumulative snapshots from the beginning of the dataset, each larger by a day, week, or year. Some divide all the data into fixed-sized nonoverlapping (or regularly overlapping) snapshots. Leskovec et al. [13] found that although the snapshots added nodes and edges over time, the graphs became denser and the diameter decreased. Their work used datasets where many EAs repeat (similar to our own) and datasets where EAs occurred once per edge (e.g., citation networks). Krings et al. [3] thoroughly analyzed a single, very clean telecommunication dataset. They found distinct patterns of node count and clustering coefficient based on interval start day and length that were well explained by normal call behavior. Comparing neighboring snapshots using 2-week intervals even found holiday weeks. Rocha et al. [14] studied four sampling techniques for a stream of EAs including varying interval size. They found that static snapshots lost some detail necessary for temporal epidemic tracking and prediction.

Temporal Networks: Researchers have analyzed "temporal networks" directly. In temporal networks, EAs are directly analyzed for graph analytics (e.g., reachability or infection spread; see [15] for a survey). However, our goal is to derive static snapshots from EA data, and would leverage static graph analytics for reachability or infection spread.

Identifying Appropriate Snapshot Intervals: Sun et al. [1] proposed a mostly parameter-free technique called GraphScope for tracking graph communities in EA streams. They batch all edges that advertise during a fixed interval (the one parameter) into a graph, compute its communities based on small compressed size, and add it to the previous snapshot "if there is a storage benefit". Otherwise, this graph starts a new snapshot. EASEE can break an EA stream into a new snapshot at any point, not just at fixed time boundaries, and uses overall graph stability measures, not communities. Sulo et al. [2] developed an off-line technique that finds a universal snapshot interval that balances within-snapshot variance and between-snapshot compression. Cáceres [4] described an off-line technique called DAPPER that allows varying snapshot intervals, but they are multiples of a fixed minimum interval. Breakpoints between snapshots are based on measures of edge frequency consistency. Soundarajan et al. [5] proposed ADAGE as an on-line technique that merges neighboring small fixed snapshots until a user-specified graph metric converges. They found that although different metrics changed final snapshot intervals, often cheaper proxy metrics provided approximately the same result as many more expensive metrics (e.g., degree distribution as a proxy for Page Rank).

Fish et al. [6] presented a technique akin to supervised learning that trains a snapshot detector to increase accuracy against a specified, dataset-specific labeled problem (e.g., change point detection; node attribute prediction). EASEE finds change points directly in the EA stream without labels.

Concurrent Research: As we were finalizing this paper for submission, Orman et al. [7] published a snapshot detecting technique. They merge neighboring, fixed-sized snapshots if the edge or node Jaccard similarity deviates enough from a previous null model. The fixed-size "shortest stable duration" (ϵ) is intended to avoid undersampling noise. They suggest testing a candidate ϵ by computing the sequence of similarities between

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3223614

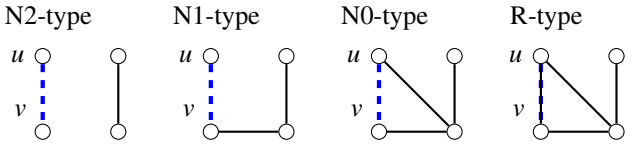


Fig. 1: The four possible types for new EA (u, v, t) denoted by the dashed line. The illustrations show a portion of the graph just prior to time t – shown as solid lines.

neighboring snapshots. It is good if both this sequence's variance and its compressibility (using measures similar to those in [2]) are low. Orman's technique is sensitive to how their parameters are selected, but do not describe how to select their sequence variance and compressibility cut-offs, nor provide default values for them. EASEE is an on-line technique that examines each EA (no fixed window) as it arrives for adaptive-sized minimal snapshot detection, and cosine similarity for our merging step. Furthermore, we show in Sec. V-C that cosine similarity performs better at this task than Jaccard similarity.

III. The EASEE Model

EASEE has two main steps: (1) Detect sufficient snapshots, and (2) Merge neighboring snapshots based on graph similarity. The first addresses undersampling while avoiding oversampling. The second permits merging (to avoid unnecessary snapshot splitting) while preserving oversampling controls.

In this section, we describe each step. First, we describe a model to predict near-term graph growth as more EAs are added to a snapshot. Second, we use that model to measure when a minimum snapshot size has converged. Third, we describe a metric that identifies which neighboring minimum snapshots may be merged with measured amounts of oversampling.

A. Forecasting near-term growth

Suppose u and v are two entities (nodes) that communicate at time t. We represent this communication by undirected edge (u, v) and denote the communication event by (u, v, t). We call each event an *edge advertisement* (EA) and say that edge (u, v) *advertises* at time t. We refer to the ith EA as $(u, v, t)_i$. Edges may advertise more than once.

First, note that each EA is one of four possible *types* (Fig. 1):

1) *N2-type*: A new edge with two previously unobserved nodes. This type creates a new connected component.
2) *N1-type*: A new edge with one previously unobserved node.
3) *N0-type*: A new edge with zero new nodes. That is, both u and v were involved in at least one previous EA, but not simultaneously. This EA can either merge two existing components or "densify" an existing component.
4) *R-type*: A repeat of an edge already seen advertised.

These are the only four types. Each EA provides exactly two nodes with exactly two possible values on each (seen before or not). Symmetry where one node was seen reduces this from four cases to three. Adding if the edge is a repeat when both nodes are seen before increases up to four cases.

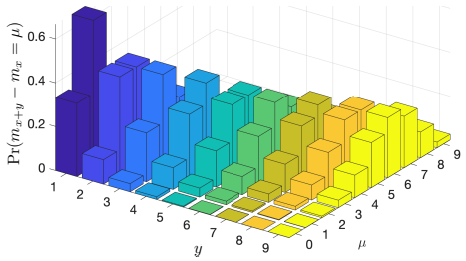


Fig. 2: The probability distribution of number of future edges $\Pr(m_{x+y} - m_x = \mu)$, assuming $y = 1, \ldots, 9$ additional edge advertisements and $p(\mathrm{R}, x) = 1/3$.

Given a stream of x EAs, we can estimate $p(T, x)$, the probability that a new EA will be of type $T \in \{\mathrm{N2}, \mathrm{N1}, \mathrm{N0}, \mathrm{R}\}$. We use this to predict the number of new edges and new nodes that will be added to the graph given $y \geq 1$ further EAs.

Number of edges: Let m_x denote the number of edges in the graph after the xth EA. The $(x+1)$st EA will be a repeat of a previous edge with probability $p(\mathrm{R}, x)$, so that

$$m_{x+1} - m_x = \begin{cases} 0 & \text{with probability } p(\mathrm{R}, x) \\ 1 & \text{with probability } \big(1 - p(\mathrm{R}, x)\big). \end{cases} \tag{1}$$

With this, we derive the probability distribution for the general case of $y \geq 1$ additional edge advertisements, that is, the distribution of $m_{x+y} - m_x$.

To do so, we make two simplifying assumptions. First, we assume $p(\mathrm{R}, x) = p(\mathrm{R}, x+1) = \cdots = p(\mathrm{R}, x+y)$ – i.e., the probability a repeat edge has "stabilized" after x advertisements and stays the same with y new advertisements. As we choose y to be relatively small when compared to x, this assumption generally holds well. Second, for simplicity, we assume the new edge advertisements are mutually independent. By these assumptions, $m_{x+y} - m_x = (m_{x+y} - m_{x+y-1}) + \cdots + (m_{x+1} - m_x)$ is the sum of y independent random variables, each with the distribution defined by Eq. (1). This is the definition of the well-known binomial distribution with y trials, each with success probability $1 - p(\mathrm{R}, x)$:

$$\Pr(m_{x+y} - m_x = \mu) = \binom{y}{\mu} p(\mathrm{R}, x)^{y-\mu} \big(1 - p(\mathrm{R}, x)\big)^{\mu} \tag{2}$$

is the probability that exactly $\mu \in \{0, 1, \ldots, y\}$ new edges will be added to the graph after y new advertisements. Thus, the mean or expected number of new edges is [16]

$$\mathbb{E}[m_{x+y} - m_x] = y \big(1 - p(\mathrm{R}, x)\big). \tag{3}$$

This expectation has an intuitive interpretation: Repeat-type EAs are the only ones which do not add a new edge to the graph. Therefore, the number of expected new edges after y EAs is y times the expected proportion for all other EA types.

For example, Fig. 2 illustrates Eq. (2) assuming $y = 1, \ldots, 9$ more EAs and $p(\mathrm{R}, x) = 1/3$. For example, the orange bars illustrate the probability that the graph will include from 0 to 8 more edges after $y = 8$ more EAs; this probability initially grows with μ and reaches a peak at $\mu = 5$, meaning that 5 additional edges is the most likely outcome.

Number of nodes: Let n_x denote the number of nodes in the

graph after the xth EA. The $(x+1)$st EA can add zero, one, or two new nodes following a categorical distribution with

$$n_{x+1} - n_x = \begin{cases} 0 & \text{with probability } p(\mathrm{R},x) + p(\mathrm{N0},x) \\ 1 & \text{with probability } p(\mathrm{N1},x) \\ 2 & \text{with probability } p(\mathrm{N2},x). \end{cases} \quad (4)$$

With this, we derive the probability distribution $\Pr(n_{x+y} - n_x)$ for the general case of $y \geq 1$ more EAs. We make the same simplifying assumptions discussed for edges (above) but we also assume that $p(\mathrm{N0},x)$, $p(\mathrm{N1},x)$, and $p(\mathrm{N2},x)$ have also stabilized to remain constant for $x+1,\ldots,x+y$. Under these assumptions, $n_{x+y} - n_x$ is the sum of y iid categorical random variables, where each takes one of three values with probabilities defined by Eq. (4). Using the multinomial theorem [17], we can show that

$$\Pr(n_{x+y} - n_x = \nu) = \sum_{i=\max(0,\nu-y)}^{\lfloor \nu/2 \rfloor} \binom{y}{i+y-\nu, \nu-2i, i} \times \left(p(\mathrm{R},x) + p(\mathrm{N0},x)\right)^{i+y-\nu} p(\mathrm{N1},x)^{\nu-2i}\, p(\mathrm{N2},x)^{i} \quad (5)$$

is the probability that ν new nodes will be added to the graph following $y \geq 1$ new EAs, where $\nu \in \{0,1,\ldots 2y\}$, and

$$\binom{n}{a,b,c} = \frac{n!}{a!\,b!\,c!}$$

is a multinomial coefficient. For example, by Eq. (5), the probability of $\nu = 2$ and $\nu = 5$ new nodes after $y = 3$ additional edge advertisements is

$$\Pr(n_{x+3} - n_x = 2) = 3\left(p(\mathrm{R},x) + p(\mathrm{N0},x)\right) p(\mathrm{N1},x)^2 + 3\left(p(\mathrm{R},x) + p(\mathrm{N0},x)\right)^2 p(\mathrm{N2},x)$$
$$\Pr(n_{x+3} - n_x = 5) = 3\, p(\mathrm{N1},x)\, p(\mathrm{N2},x)^2.$$

The expected number of new nodes is simply y times the mean value of $n_{x+1} - n_x$, that is,

$$\mathbb{E}[n_{x+y} - n_x] = y\left(p(\mathrm{N1},x) + 2\,p(\mathrm{N2},x)\right). \quad (6)$$

Intuitively, this says that N1-type EAs add one node, N2-type EAs add two nodes, and other types add none. The expectation after y EAs is y times the weighted expected proportion.

Fig. 3 illustrates Eq. (5) assuming $y = 1,\ldots,9$ additional edge advertisements and parameters $p(\mathrm{R},x) + p(\mathrm{N0},x) = 0.33$, $p(\mathrm{N1},x) = 0.25$, and $p(\mathrm{N2},x) = 0.42$. For example, the orange bars illustrate the probability that the graph will include anywhere from 0 to $2y = 16$ more nodes after $y = 8$ more EAs; this probability initially grows with ν and reaches a peak at $\nu = 9$: Nine additional nodes is the most likely outcome.

B. Detecting sufficient snapshots

EASEE's first step creates minimum-sized static graphs from the EA stream. Thus, it must ensure enough data has arrived that undersampling will be minimized, but (to avoid oversampling) take care that no significant changes occur.

Specifically, EASEE partitions the EA stream into non-overlapping adjacent intervals, specified by interval size and right endpoint, and builds a snapshot on each interval. Let $w_1, w_2, \ldots$ be a sequence of interval sizes, and define

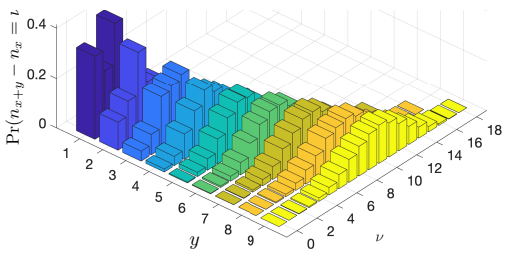


Fig. 3: The probability distribution of future nodes $\Pr(n_{x+y} - n_x = \nu)$, assuming $y = 1,\ldots,9$ additional edge advertisements and parameters $p(\mathrm{R},x) + p(\mathrm{N0},x) = 0.33$, $p(\mathrm{N1},x) = 0.25$, and $p(\mathrm{N2},x) = 0.42$.

$r_j = \sum_{i=1}^{j} w_i$ to be the right endpoint of the jth interval. By convention $r_0 \equiv 0$. Then the edge and node sets for the jth snapshot in the sequence are

$$\mathcal{E}_j = \{(u,v)\colon (u,v,t)_i, i = r_{j-1}+1,\ldots,r_j\}, \text{ and}$$
$$\mathcal{V}_j = \{u\colon (u,v) \in \mathcal{E}_j \text{ or } (v,u) \in \mathcal{E}_j\}. \quad (7)$$

$(\mathcal{V}_1,\mathcal{E}_1)$ is a snapshot defined by the first w_1 EAs, $(\mathcal{V}_2,\mathcal{E}_2)$ is a snapshot defined by EAs $w_1+1,\ldots,w_1+w_2$, etc.

The expectations defined by Eqs. (3) and (6) are large at the beginning of a snapshot interval, but both rapidly decrease with increasing x (the number of EAs). Thus, early EAs are more likely to be N2 type, but eventually a steady proportion become R type. A snapshot interval is sufficient when the expected number of new nodes and edges converges. Specifically, an interval ends when the smoothed prediction trends start to increase again – indicating convergence (the derivative is zero) or a non-trivial change in the underlying data stream (the derivative is greater than zero).

EASEE's initial output is a series of snapshots $(\mathcal{V}_1,\mathcal{E}_1)$, $(\mathcal{V}_2,\mathcal{E}_2),\ldots$ with sufficient intervals. These intervals $\{w_i\}$ are intentionally kept as small as possible to limit the effect of underlying data change. Alg. 1 describes this step – excepting a few settings derived from real data (see Secs. V-A and V-B).

C. Merging neighboring snapshots

When creating a static snapshot representation from dynamic, temporal data, creating a static snapshot across catastrophic underlying changes leads to oversampling. Step one avoids this by splitting when the derivative is non-negative. However, sometimes neighboring sufficient snapshots are highly similar, and the resulting graph snapshot stream would be improved by merging these snapshots. To ensure such snapshots aren't merged across oversampling changes, EASEE quantifies the amount of change between two in-time consecutive snapshots – merging only if the change is small.

Specifically, EASEE uses the cosine similarity of the advertisement rate between adjacent sufficient snapshots, i.e.,

$$\frac{c_j \cdot c_{j+1}}{||c_j||_2\, ||c_{j+1}||_2}, \quad (8)$$

where c_j is a vector of either EA counts (number of times an edge advertises in snapshot j) or node EA-participation counts (number of times a node is in an EA in snapshot j). Cosine similarity values range from 0 (no similarity) to 1 (equal).

Algorithm 1: Identify Sufficient Snapshots

Data: A stream of EAs $(u, v, t)_i$, a prediction length y
Result: A stream of graph snapshots $(\mathcal{V}_j, \mathcal{E}_j, w_j)$

$w_j \leftarrow 0$; numAds $\leftarrow 0$; $\mathcal{V}_j \leftarrow$ countingSet(); $\mathcal{E}_j \leftarrow$ countingSet(); lastAds $\leftarrow 0$; RecentEAs $\leftarrow$ list()
while *EA stream continues* **do**
  $(u, v, t) \leftarrow$ getNextEA()
  if $(u, v) \in \mathcal{E}_j$ **then**
    type $\leftarrow R$
  else if $u \in \mathcal{V}_j$ *AND* $v \in \mathcal{V}_j$ **then**
    type $\leftarrow N0$
  else if $u \in \mathcal{V}_j$ *OR* $v \in \mathcal{V}_j$ **then**
    type $\leftarrow N1$
  else
    type $\leftarrow N2$
  end
  $\mathcal{E}_j$.add((u, v)); $\mathcal{V}_j$.add(u); $\mathcal{V}_j$.add(v)
  numAds + = 1
  RecentEAs.append(type)
  `/* Fix RecentEAs size: Sec.V-A. */`
  AdProbs $\leftarrow$ percentsByType(RecentEAs)
  $\mathbb{E}[m_{\text{numAds}+y}], \mathbb{E}[n_{\text{numAds}+y}] \leftarrow$ Eqs. (3), (6)
  `/* Convergence details: Sec.V-B. */`
  if *converged*($\mathbb{E}[n_{numAds+y}], \mathbb{E}[m_{numAds+y}]$) **then**
    $w_j \leftarrow$ numAds − lastAds
    lastAds = numAds
    releaseSufficientSnapshot($\mathcal{V}_j$, $\mathcal{E}_j$, w_j)
    $\mathcal{V}_j$.clear(); $\mathcal{E}_j$.clear(); RecentEAs.clear()
end

Algorithm 2: Merge Neighboring Snapshots

Data: A stream of graph snapshots $(\mathcal{V}_j, \mathcal{E}_j, w_j)$, threshold t_c
Result: A stream of merged graph snapshots $(\mathcal{V}_{\text{out}}, \mathcal{E}_{\text{out}}, w_{\text{out}})$

$\mathcal{V}_{\text{active}} \leftarrow \emptyset$; $\mathcal{E}_{\text{active}} \leftarrow \emptyset$; $w_{\text{out}} \leftarrow 0$
while *graph stream continues* **do**
  $(\mathcal{V}_j, \mathcal{E}_j, w_j) \leftarrow$ getNextSufficientSnapshot()
  mergeGraph $\leftarrow$ True
  for $\mathcal{V}_z \in \mathcal{V}_{active}$ *AND* $\mathcal{E}_z \in \mathcal{E}_{active}$ **do**
    nodeSim $\leftarrow$ Eq. 8($\mathcal{V}_j$, $\mathcal{V}_z$)
    edgeSim $\leftarrow$ Eq. 8($\mathcal{E}_j$, $\mathcal{E}_z$)
    if *nodeSim* $< t_c$ *OR edgeSim* $< t_c$ **then**
      mergeGraph $\leftarrow$ False
  end
  if *mergeGraph == False* **then**
    $\mathcal{V}_{\text{out}} \leftarrow$ unionAll($\mathcal{V}_{\text{active}}$); $\mathcal{E}_{\text{out}} \leftarrow$ unionAll($\mathcal{E}_{\text{active}}$)
    releaseMergedGraph($\mathcal{V}_{\text{out}}$, $\mathcal{E}_{\text{out}}$, w_{out})
    $\mathcal{V}_{\text{active}} \leftarrow \emptyset$; $\mathcal{E}_{\text{active}} \leftarrow \emptyset$; $w_{\text{out}} \leftarrow 0$
  $\mathcal{E}_{\text{active}}$.add($\mathcal{E}_j$); $\mathcal{V}_{\text{active}}$.add($\mathcal{V}_j$); w_{out}+ = w_j
end

If both measures' values are at least a threshold t_c, then EASEE merges the two snapshots. Before merging a third (or more) snapshot(s) to an earlier set, EASEE ensures all paired cosine similarities are above the threshold. See Alg. 2.

In Sec. V-C, we show why cosine similarity is better than Jaccard or weighted Jaccard similarity and provide a heuristic for choosing t_c against real data.

D. EASEE Parameters Overview

EASEE requires three parameters: a recent history size (x) for estimating EA-type probabilities ($p(T, x)$); a smoothing window size for identifying sufficient interval convergence; and the merging threshold (t_c). After introducing our real-world datasets (Sec. IV), we show in Sec. V how we selected values for these parameters from samples of the data. In brief, we show that the recent history size is rather consistent across our varying datasets – selecting a single value for all of them. We show that the smoothing window size changes results linearly, and select a single value that works well across all datasets. We show how to select an appropriate merging threshold for each dataset by noting how sweeping the value alters the number of merged snapshots – identifying a clear cut-off point.

IV. Data

To find appropriate settings for our parameters (Sec. V), we applied EASEE to seven datasets. We describe those datasets here before analyzing the parameters. See Table I for data sizes. To simulate streaming data, we sorted all EAs by increasing timestamp. Finally, we removed duplicate EAs – the same source, destination, and time. This last step cleaned several datasets that had false repeats (e.g., the same emails in several Enron inboxes). Although we preserved communication direction in the data, our EASEE implementation considers graphs undirected and thus ignores direction.

Our Python code finished in far less time than the interval time represented in each dataset. Our code reads each EA from file, updates internal state, and identifies sufficient snapshots before continuing to the next EA. Thus, the algorithm and code do not require massive data interactions at any one time. Thus, EASEE processes data far faster than the average arrival rate for any of our datasets. The shortest period – also the fastest arrival rate – for any of our datasets was LANL Netflow with more than 18 days' worth of data. Our single threaded, unoptimized Python code completed in 32 hours, 5 minutes on a Mac Pro with 3.7 GHz Intel Xeon E5 and 64GB RAM. All other datasets represent much longer periods, and the analyses completed in far less time. Several datasets contain millions of nodes and edges, and in one case, billions of EAs.

EU Emails: Email represents a common, often repeating communication type used to build static graphs [18].[1] This dataset presents EAs for each email sent between members of a European research institution – but not for those where either participant is outside that institution.

Enron Emails: This dataset provides all emails and directory structure for around 150 users at the Enron Corporation.[2] We generated separate EAs for each address in the "to", "cc" and "bcc" fields of each email.

[1] https://snap.stanford.edu/data/email-Eu-core-temporal.html
[2] https://www.cs.cmu.edu/~./enron/

TABLE I: Some properties for our datasets.

Name	Number of EAs	Number of Nodes	Number of Edges
EU Core Emails	327,336	986	16,064
Enron Emails	1,283,755	84,511	316,061
GameX Combats	500,327	10,589	86,351
GameX Messages	4,515,396	22,442	293,860
Stack Overflow Reply	63,496,479	2,601,977	29,541,284
Reddit Reply	646,024,723	8,901,033	437,747,667
LANL Netflow	2,585,934,400	166,925	1,237,992

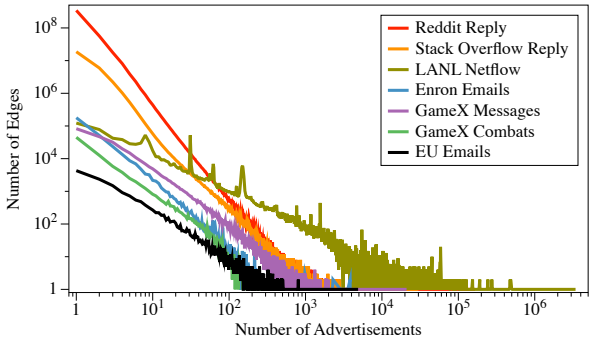


Fig. 4: The EA distribution for all datasets. Although a significant number of edges advertised only once, a non-trivial number advertised hundreds to thousands of times.

GameX Messages and Combat: "GameX" is a browser-based game. Players travel/explore a fictional world. They can mine resources, trade, and conduct war. They can communicate through in-game personal messages. The GameX Message and GameX Combat datasets are logs of messages and combat events (respectively) between anonymized players.[3]

Stack Overflow Reply: Stack Overflow is a question-and-answer site. This dataset contains an EA for each time a user posted to another user's question or answer [18].[4]

Reddit Reply: Reddit is a social platform where users submit and discuss news content. EAs exist for when any user commented on another user's post or comment [19], [20].[5]

LANL Netflow: We used the Time, SrcDevice, and DstDevice columns from Los Alamos National Laboratory's Unified Host and Network dataset [21].[6] We used the first 32 days of data, which contains EAs generated by human and automated computer actions.

Fig. 4 shows the EA distribution for each dataset. While many edges advertise only once, many produce hundreds to thousands of EAs: These are long-tailed distributions.

V. Model Calibration From Data

Sec. III describes the EASEE model. The user must set three parameters: recent history size, smoothing window size, and merging threshold. In this section, we show how to use real data to select an appropriate recent history size and demonstrate that EASEE is robust to a wide variety of values for this parameter. We briefly discuss the second parameter. We provide an analysis of the third and show that cosine similarity performs better than Jaccard or weighted Jaccard similarities.

[3]To maintain full player privacy, we are not permitted to share this data nor provide further identifying details for which game generated the data.

[4]https://snap.stanford.edu/data/sx-stackoverflow.html

[5]Jason Baumgartner, pushshift.io. https://www.cs.cornell.edu/~arb/data/temporal-reddit-reply/

[6]https://csr.lanl.gov/data/2017.html

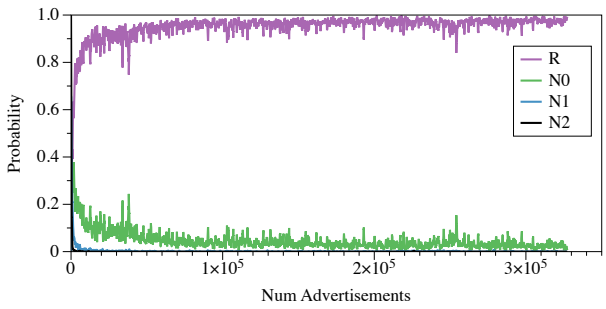


Fig. 5: EA type probabilities for the EU Email dataset with recent history size set to 1,000.

A. Probabilities from Recent History

We can estimate the probability of the four EA types – N2, N1, N0, R (Sec. III-A and III-B) – from their fractions of the most recent EAs. Here, we show how to identify a suitable recent-history size. We also show that this size parameter is robust to a wide range of values across a wide range of datasets.

Fig. 5 illustrates probability estimates for each EA type for the EU Email dataset for one recent-history size. It shows that although there is some noise in the estimates, the approximate values are surprisingly stable: $p(\text{N2})$ converges rapidly to near-zero; $p(\text{N1})$ converges only slightly more slowly to near-zero; $p(\text{N0})$ converges to non-zero; and $p(\text{R})$ converges to near-one. Although different datasets converge to different final values for each EA type, and some contain much more motion and noise, all show similar results.

To detect a sufficient snapshot, EASEE predicts near-future edge and node sizes (Eqs. (3) and (6)). As EASEE uses these predictions to identify steady-state convergence or sudden underlying data shift, these predictions must be accurate. Herein, we show how to select an optimal recent-history size for a fixed data set to minimize prediction error.

Given a data sample, we use the following to find the optimal recent-history size (k). For varying k: (1) Estimate $p(\text{T}, x)$, the percent of the most recent k EAs that are type T. (2) Use these estimates to compute the expected number of new nodes and edges (Eqs. (3) and (6); $y = 100$). (3) Compute the error between the expected values from Step 2 and actual change after $y = 100$ EAs. (4) Move forward one EA, update percentages, predictions, and errors; repeat for all EAs in the data sample. (5) Collect all errors and report the root mean squared error (RMSE).

Figs. 6(a)–(g) show the RMSE for all of our datasets across 24 different recent history sizes (k =10; 50; 100; 150; 250; 350; 450; 600; 850; 1,150; 1,600; 2,200; 3,050; 4,200; 5,760; 7,900; 10,850; 14,850; 20,450; 28,050; 38,550; 53,000; 72,800; 100,000). All datasets' results are similar: (1) When the recent

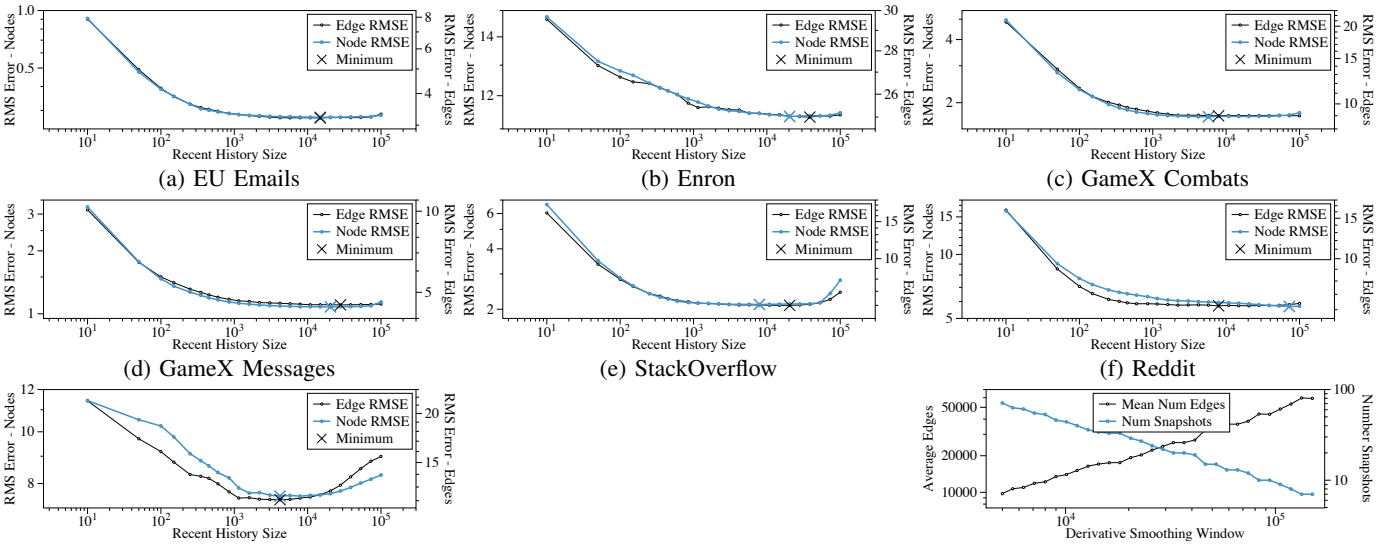


Fig. 6: (a)–(g): The root mean squared error (RMSE) for predicting the graph size for the first 300K EAs for all our datasets. (h): The effect of the prediction smoothing window size on snapshot size and number of snapshots found for the Enron dataset.

history size is small, predicted graph sizes have significant errors. (2) This error decreases, plateaus, and increases again. (3) The shapes (and minima) of the node and edge curves are surprisingly similar (Reddit Reply being the only exception). Even more surprising is the width of the plateaus and their consistency across all datasets. Even LANL Netflow with the narrowest plateau spans a full order of magnitude. Therefore, for simplicity in all following experiments, we used a fixed recent history size within all plateaus – $k = 5,000$ EAs.

This analysis could be repeated off-line for any dataset sample. Although an on-line analysis could be performed, we didn't develop such a technique. We feel across-dataset consistency indicates this parameter is stable. We prefer a simpler algorithm over one that could adapt to unknown other datasets that could possibly be different on-line.

B. Identifying Sufficient Snapshots

We end a sufficient interval when growth stabilizes or suddenly increases (i.e., when smoothed prediction trends are flat or increase).

We want both node and edge growth to be stable. Therefore, a sufficient interval ends when the smoothed derivatives of both nodes and edges reach or exceed zero. Without noise, finding this endpoint would be trivial. However, the predictions have considerable noise – from recent history size and data motion. Furthermore, numerical derivatives *increase* noise. Therefore, we smooth the numerical derivatives over a window.

We tested how different smoothing window sizes affect sufficient intervals. Fig. 6(h) shows how the smoothing parameter affects the number of snapshots generated (blue; right y-axis), and the size of each snapshot (black; left y-axis) for Enron. We saw the same trend for all datasets: A larger smoothing window led to larger graphs and fewer snapshots. Thus, the larger the smoothing window, the more likely EASEE will miss changes in the communication stream. After some manual evaluation, and as the analysis in Fig. 6(h) showed no knee in the data, we selected a value that worked well across all datasets – 10,000. This value resulted in well-sized snapshots for all datasets. Furthermore, with no reason to choose from any of the various values, we chose a round number.

C. Merging Threshold and Metric

The final parameter is the merging threshold t_c for the snapshot similarity metric (Sec. III-C). We propose pairwise cosine similarity as the similarity metric (Eq. 8). In this section, we first demonstrate that Jaccard, weighted Jaccard, and cosine similarities approximate the ℓ_0 pseudo-norm and the ℓ_1, and ℓ_2 norms, respectively. We then show that cosine similarity better leverages its range and provides better differentiation to help identify clear values for the threshold parameter t_c.

Similarity metric definitions: Recall that the simplest representation of each snapshot is as a list of edges and nodes with their corresponding EA counts. (For nodes, we use the number of EAs the node appears in.) Thus, edges and nodes may be represented by a vector of EA counts, where each entry in the vector corresponds to a particular edge or node. There are three common metrics for comparing graphs represented thus – Jaccard, weighted Jaccard, and cosine similarities [22].

Let c_j and c_{j+1} be edge or node vectors for neighboring snapshots with non-negative entries $c_{j,k}$ and $c_{(j+1),k}$ (the count for edge or node k). Jaccard similarity is defined as

$$\frac{\left|\{c_{j,k} \neq 0\} \cap \{c_{(j+1),k} \neq 0\}\right|}{\left|\{c_{j,k} \neq 0\} \cup \{c_{(j+1),k} \neq 0\}\right|}. \tag{9}$$

where $|\cdot|$ denotes set cardinality, $\cap$ set intersection, and $\cup$ set union. Weighted Jaccard similarity is defined as

$$\frac{\sum_k \min\{c_{j,k}, c_{(j+1),k}\}}{\sum_k \max\{c_{j,k}, c_{(j+1),k}\}}. \tag{10}$$

Cosine similarity was already defined in Eq. (8).

We can re-write Jaccard similarity as:

$$\frac{\left\|c_j \odot c_{j+1}\right\|_0}{\left\|c_j\right\|_0 + \left\|c_{j+1}\right\|_0 - \left\|c_j \odot c_{j+1}\right\|_0}, \tag{11}$$

where $\odot$ denotes the Hadamard or element-wise product, and $\|\cdot\|_0$ denotes the ℓ_0 pseudo-norm (that counts the number of non-zero entries). Weighted Jaccard similarity becomes

$$\frac{\left\|c_j\right\|_1 + \left\|c_{j+1}\right\|_1 - \left\|c_j - c_{j+1}\right\|_1}{\left\|c_j\right\|_1 + \left\|c_{j+1}\right\|_1 + \left\|c_j - c_{j+1}\right\|_1}, \tag{12}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm. Cosine similarity was already written in terms of the ℓ_2 norm.

This shows a rough correspondence between the Jaccard, weighted Jaccard, and cosine similarities and ℓ_0, ℓ_1, and ℓ_2 norms respectively. Thus, choosing between Jaccard, weighted Jaccard, and cosine similarities is akin to choosing between these standard norms – with similar expectations. Jaccard counts simple edge or node existence in both snapshots with the same weight independent of how many times it advertised in each snapshot. Weighted Jaccard increases the importance of having similar counts. By having a squared-like property, cosine similarity increases the importance of high-count edges or nodes proportionally higher than low-count edges.

Comparing similarity metric results: Recall that our goal is to identify whether neighboring sufficient snapshots can be merged without increasing oversampling. Herein, we demonstrate both that cosine similarity is a better similarity measure, and that there is generally an identifiable point for the threshold t_c.

Given a series of sufficient snapshots $\{(\mathcal{V}_1, \mathcal{E}_1), (\mathcal{V}_2, \mathcal{E}_2), \ldots (\mathcal{V}_d, \mathcal{E}_d)\}$, we form a $d \times d$ similarity matrix, where the (r, s) entry denotes the similarity between snapshots r and s. We then perform a sensitivity analysis by (1) sorting all unique values in the matrix in increasing order, (2) sweeping the merging threshold over these values, and (3) plotting the number of resulting merged snapshots. Similar to choosing the number of clusters or principal components from a scree plot [23], we look for a knee in the plot. Much like selecting the recent history size k (Sec. V-A), this method is not entirely online: it requires enough data for many sufficient snapshots. A fully online or adaptive method remains unsolved.

Fig. 7 demonstrates the results for all of our datasets for this analysis. To make the computations feasible for our larger datasets, we use only the first 200 minimum snapshots (from Alg. 1).As stated above, this analysis is intended to be performed on a subset of the data. Note that the node similarities are generally higher than their corresponding edge values – requiring higher thresholds to result in the same number of snapshots. Thus, we consider only the edge similarities going forward. Next, notice that Jaccard and weighted Jaccard provide surprisingly similar scores for the edge values. Furthermore, the Jaccard and weighted Jaccard scores (1) are much lower than the cosine scores, (2) use less of the [0, 1] range available to these metrics, and (3) generally show less significant knees than the cosine similarities.

Thus, we use cosine similarity on the edges for determining which minimal snapshots to merge in EASEE Step 2. As selecting knees on scree plots is an inexact technique (and specific choices could be affected by down-stream application needs), we selected our own in a few plots that follow, but do not indicate specific values on each of the plots in Fig 7.

VI. Experiments

In this section, we thoroughly analyze EASEE against both known-answer, synthetic datasets and several real-world datasets. Each subsection covers a different analysis: Sec. VI-A shows how well EASEE finds changes with varying rates of underlying temporal change against known-answer, synthetic data; and shows that a recent technique fails. Sec. VI-B shows how important it is to find such changes to downstream analytics: EASEE's change detection enables community detection against the resulting snapshots, while missing the change fails for community detection (even if the exact right *number* of EAs are combined in snapshots). Sec. VI-C shows that EASEE quickly converges to stable snapshots against real-world data – even when started at very different times. Sec. VI-D shows EASEE's merge results against our real-world datasets – uncovering results not seen before in any previous works. Sec. VI-E discusses how EASEE can identify EA datasets that should not be used to create static snapshots.[7]

A. Synthetic Data 1: Varying Change Rate Detection

For this test, we measure how responsive EASEE's first step is to identifying changes in the underlying communication streams. There are three factors that affect the ability of EASEE to detect these changes: (1) *Change amount*: As shown in Fig. 5, some new edges are joining at all times. How many new edges and nodes are needed to find a change point? (2) *Change rate*: If a change is spread across a long period, it becomes a series of small changes. How rapid a change is needed to find a change point? (3) *EASEE convergence*: As shown on the left portion of Fig. 5, EASEE must collect some amount of data before the EA type probabilities become nearly-stable. How does decreased convergence time before the change affect EASEE's ability to find it?

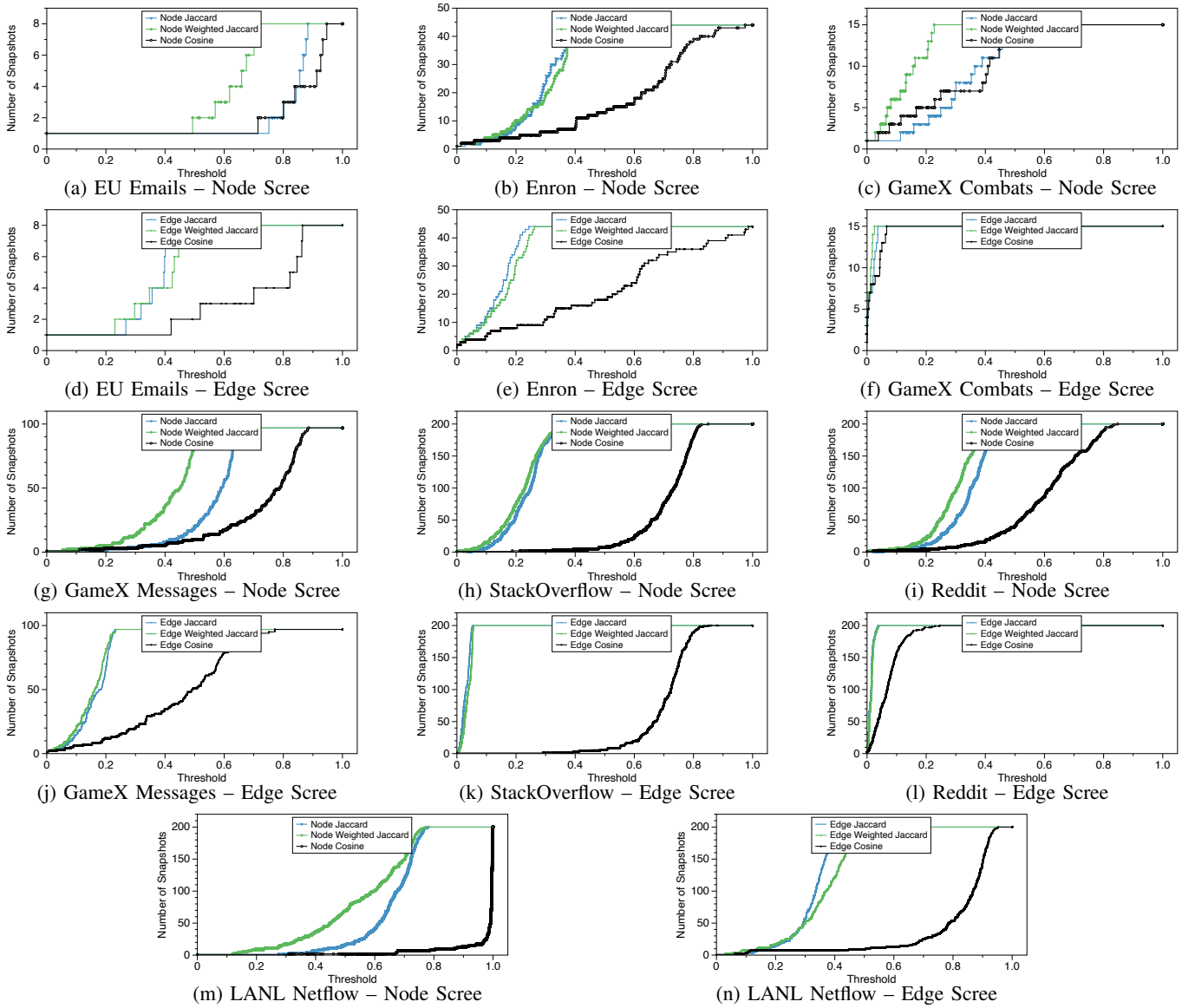
Synthetic EA generation: We developed a dataset of synthetic EAs with a single known transition between two different periods of activity. Let G be an undirected static graph with n nodes built from the stochastic block model (SBM) with 10 blocks, and let $A = \{a_{i,j}\}$ denote the $n \times n$ adjacency matrix of G. Specifically, we use an SBM where diagonal blocks have connection probability p_{in} and off-diagonal blocks have connection probability p_{out}. (Although the SBM isn't used in this analysis, it is in Sec. VI-B. To keep the two synthetic generators as similar as possible, this one uses SBM as well.)

To handle the first factor (change amount), let

$$A_0 = \left\{a_{ij} : r \le i, j \le r + \frac{n}{2}\right\}$$
$$A_1 = \left\{a_{ij} : \frac{n}{2} - r \le i, j \le n - r\right\}$$

denote two submatrices of A, where $r \in \{0, 1, ..., n/4\}$ is an integer parameter (to simplify notation, we assume n is divisible by 4). By construction, A_0 and A_1 are defined by the $n/2 \times n/2$

[7]Our code is published at https://github.com/sandialabs/Easee/.



(a) EU Emails – Node Scree (b) Enron – Node Scree (c) GameX Combats – Node Scree

(d) EU Emails – Edge Scree (e) Enron – Edge Scree (f) GameX Combats – Edge Scree

(g) GameX Messages – Node Scree (h) StackOverflow – Node Scree (i) Reddit – Node Scree

(j) GameX Messages – Edge Scree (k) StackOverflow – Edge Scree (l) Reddit – Edge Scree

(m) LANL Netflow – Node Scree (n) LANL Netflow – Edge Scree

Fig. 7: Scree plots for the node and edge similarity metrics. Although generally used for identifying knees in PCA analysis, we use it here to find similar knees in cosine similarity threshold (x-axis) vs. number of merged snapshots (y-axis).

top left and bottom right submatrices of A, respectively, offset by r; this is illustrated by Fig. 8.

We can interpret A_0 and A_1 as adjacency matrices for two subgraphs of G. These two graphs have edges and nodes in common for $r > 0$; in particular, they share $2r$ nodes. Thus, r leads to a change amount metric, defined as the ratio of shared area in the adjacency matrix in each subgraph, i.e.,

$$\text{overlap} = \frac{(2r)^2}{\left(\frac{n}{2}\right)^2} = \frac{16r^2}{n^2}. \tag{13}$$

We note that overlap = 0 and 100% for the special cases of $r = 0$ and $r = n/4$, respectively.

Let $\mathcal{E}_0$ and $\mathcal{E}_1$ denote the (possibly intersecting) edge sets of the two graphs defined by A_0 and A_1. We generate a stream of EAs from these edge sets as follows. First, all EAs occur at times modeled by a homogeneous Poisson process with intensity λ. This implies that EAs occur at an average rate of λ per unit time.

To handle the second factor (change rate), for all times t, we build EAs from edge set $\mathcal{E}_0$ with probability $p_0(t)$ and from edge set $\mathcal{E}_1$ with probability $1 - p_0(t)$, where

$$p_0(t) = \begin{cases} 1 & t \le t_0 \\ \dfrac{t - t_1}{t_0 - t_1} & t_0 < t \le t_1 \\ 0 & t > t_1 \end{cases} \tag{14}$$

and $0 < t_0 < t_1$ denote two "transition times" parameters. Eq. (14) implies that we build EAs only from $\mathcal{E}_0$ and $\mathcal{E}_1$ for times $t \le t_0$ and $t > t_1$, respectively. The EAs are drawn from both $\mathcal{E}_0$ and $\mathcal{E}_1$ at times $t_0 < t \le t_1$ (the transition zone). The probability of drawing edges from $\mathcal{E}_1$ is small but nonzero near the beginning of the transition zone; increasing linearly and reaching one at the end of the transition zone. Fig. 9 illustrates

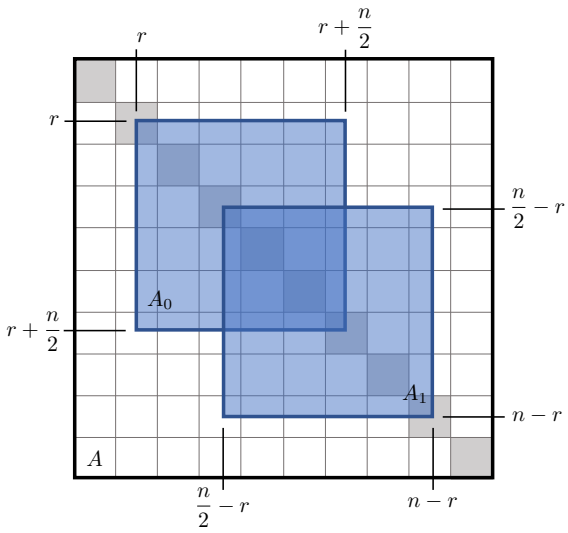


Fig. 8: Overlapping adjacency matrices A_0 and A_1 defined from a stochastic block model with 10 blocks; parameter $0 \leq r \leq n/4$ controls the amount of overlap.

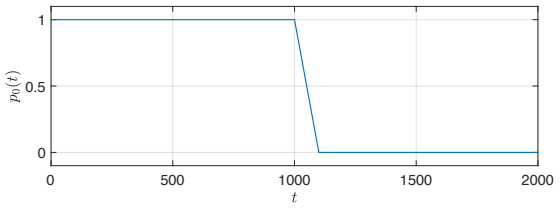


Fig. 9: Probability $p_0(t)$ of building an edge at time t using edge set $\mathcal{E}_0$. The transition zone is $[t_0, t_1] = [1000, 1100]$; the rate of transition, α defined by Eq. (15), is equal to 5%.

$p_0(t)$ assuming $t_0 = 1000$ and $t_1 = 1100$.

Suppose we generate m total EAs; the expected last EA then occurs at time m/λ. It follows that

$$\alpha = 100\,\frac{\lambda(t_1 - t_0)}{m} \tag{15}$$

quantifies the average length of the transition zone as a percentage of the total simulation time and can therefore be interpreted as a measure of the rate of transition.

The third factor (EASEE convergence) is quantified by the value of t_0 relative to the number of EAs EASEE would usually require to converge – independent of underlying data change. That is, if an infinite number of EAs were generated for edges from $\mathcal{E}_0$, what would be the average number of EAs EASEE needs to find converged snapshots? When run on millions of EAs generated from only A_0 (tested on 200 different EA streams), EASEE converged with exactly zero derivatives after between 57,068 and 90,843 EAs. Thus, using t_0 near the minimum observed value (57,068) leads to the easiest dataset – change detection near convergence; t_0 far earlier than that is a much harder dataset – change detection when far less converged.

Change detection and comparison with ADAGE: For these experiments, we used the following parameters to test these three EASEE factors (see Fig. 10): (1) *Change amount*: We used overlaps of 0%, 25%, 50%, and 75% (horizontal axis, each subfigure). (2) *Change rate*: We set the change rate to 0%, 5%, and 30% of the simulation time for immediate, medium, and slow transition (purple, green, and blue whisker plots, respectively). (3) *EASEE convergence*: We selected a near convergence $t_0 = 40,000$ – well less than the minimum natural convergence we saw. We used $t_0 = 25,000$ for a harder, not very converged case; $t_0 = 15,000$ for a hardest, recently started case. Each subfigure is labeled as one of these cases.

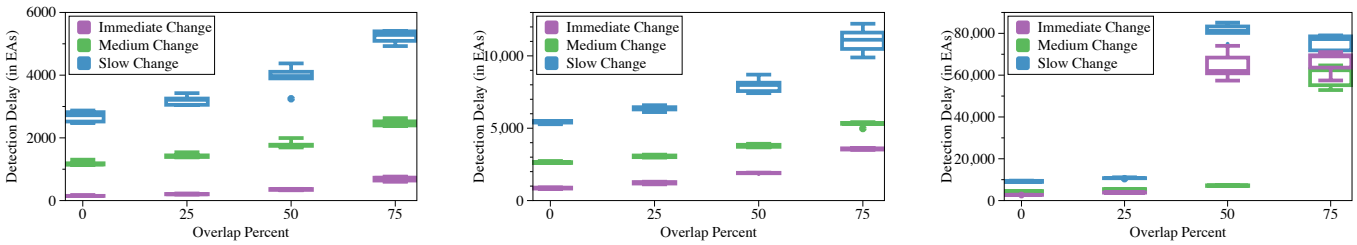
We generated datasets of 100,000 EAs for each of these parameter settings – repeated five times with different random seeds. We ran EASEE sufficient snapshot detection against each. We collected the EA number that EASEE found a snapshot after the change point for each case – always the first snapshot detected. Error was defined as the number of EAs after the transition period began until a sufficient interval was detected.

Fig. 10 shows all test results – hereafter referred to as (a), (b) or (c). When the change is immediate and the probabilities are largely converged, changes are rapidly detected ((a), purple). Even with slower changes, changes are detected relatively quickly ((a), green and blue). For each of these, there is some degradation caused by more node overlap – fewer new edges arriving after the change begins. When convergence is only partially accomplished, either the node overlap needs be very low ((b), blue, overlap=0 or 25), or the temporal change needs be relatively quick ((b), purple or green) for changes to be found around 3,000 EAs. When the sufficient interval is recently started, if temporal changes are relatively quick and the node overlap is low changes are well detected ((c), purple, green, overlap=0 or 25). However, when the node overlap is too low, all transitions fail ((c), overlap=75).

An interesting property arises in Fig. 10(c) for overlap=50 when the immediate change (purple) does far poorer than the medium rate of change (green). We believe this is because the change point is early enough that the probabilities are decreasing, so this sudden influx of some new edges is lost in the overall decreasing. However, the slower change (green) delays and spreads the influx just enough so that EASEE's measured probabilities converge enough to detect the slightly slower/delayed new edges. The slow change (blue) is slow enough that it's never detected.

To compare EASEE against prior work, we ran ADAGE against these same datasets [5]. We selected ADAGE as it was the most recent, fully automated snapshot detecting technique. ADAGE was designed to identify when sufficient EAs have been included that a social network has converged, not to identify when the communication data changes.

ADAGE requires as input a series of very small pieces (usually minutes, hours, or days), so we split the generated data into 100 EA pieces. We ran ADAGE looking for converged degree distribution. (We tested with converged clustering coefficient, but as the data is an SBM (locally Erdős-Rényi), the clustering coefficient is practically zero. The score converges almost immediately – leading to many tiny ADAGE interval sizes.) ADAGE converged to 1, 2, or 3 static graphs in each 100,000 EA dataset. We computed error similarly as for EASEE – However, we generously used the first ADAGE-found snapshot



(a) $t_0 = 40,000$ (b) $t_0 = 25,000$ (c) $t_0 = 15,000$

Fig. 10: Change detection error under varying known-truth cases. We tested our change detection against several synthetic, known-truth datasets. t_0 indicates when the transition period began for (a) nearing natural convergence ($t_0 = 40,000$), (b) farther from converged ($t_0 = 25,000$), and (c) recently started ($t_0 = 15,000$). The edge overlap between the graphs A_0 and A_1 is shown on the horizontal axis. The change rate between the two graphs is shown by the colors. Note that the vertical axis range changes for each sub-plot. See text for more explanation.

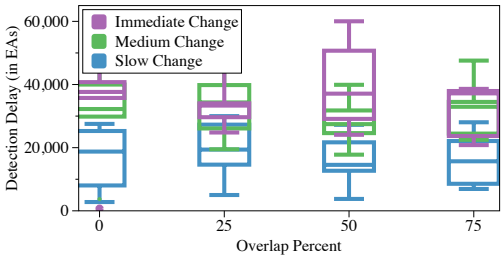


Fig. 11: Change detection error for ADAGE under the easiest known-truth case (similar to Fig. 10(a)). ADAGE delays are wildly varying, generally quite poor, and show no patterns to matching to easier or harder datasets. Although only shown for nearing natural convergence ($t_0 = 40,000$), no ADAGE results looked very different.

after t_0. Fig. 11 shows results from ADAGE against the easiest settings (similar to Fig. 10(a)). ADAGE does not appear to find any changes directly: Its results do not respond to any changes in the data difficulty. ADAGE does not find t_0 more quickly with smaller node overlap, or with faster changes. Although not shown, ADAGE's results for more difficult t_0 were similarly poor. Thus, if your goal is generating snapshots that identify drastic changes, EASEE identifies them; ADAGE does not.

B. Community detection

We next quantified the degree to which down-stream applications are affected by correctly identifying change moments in the EA stream. We selected community detection as the down-stream application.

Synthetic EA generation: We leveraged a similar process to the previous EA generation, with a few changes specific to the needs for community detection. First, for there to be truly known-answer communities, the two graphs can't ever advertise together in the same period. Otherwise, these period could create unknown communities. Thus, we use immediate transition ($t_1 = t_0 + 1$; $t_0 = 49,999$). Second, as we wanted to use Adjusted Mutual Information (AMI) [24] to measure community detection accuracy, and as EASEE's first snapshot will have some EAs from the second graph, we needed the node sets to be identical. (AMI requires there to be a correct community for all nodes in the graph. If there were nodes included in the first EASEE snapshot from EAs for the second true graph, there would be no communities for them in the correct answer for the first graph: There would be no correct AMI answer for them.) This required changes to how the SBMs were defined – described below. Third, we needed to ensure that all edges advertised at least once.This required changes to how EAs are selected – described below.

We created two SBM edge sets on the same set of nodes as follows. We want both graphs to have all nodes in common, but have different edges and communities. We created edge set $\mathcal{E}_0$ as described in Sec. VI-A, but with the requirement that all nodes have at least three neighbors: We achieved this by combining the SBM with a configuration model where each node had desired degree 3 [25].[8] This ensures that all nodes advertise enough to be seen by snapshots: Nodes with only one edge that advertises only once could be missed if the only advertisement occurred in the period when EASEE was accidentally including $\mathcal{E}_1$ edges due to latency. Adding this low-degree graph made a minimal change to the SBM probabilities in our tests. We generated edge set $\mathcal{E}_1$ the same way, except we randomly permuted the nodes. Thus, the two graphs define independent communities.

We generated a stream of x EAs from each edge set as follows. First, we assigned each edge a fixed number of EAs so that (1) each edge advertises at least once, and (2) the EA distribution is long-tailed (as for our real datasets; Fig. 4). We did this by fitting a power law distribution

$$\text{cnt}(e_i) = \text{floor}(i^a/z), \tag{16}$$

where $\text{cnt}(e_i)$ denotes the number of times the i-th randomly ordered edge advertises, m is the number of edges in the graph, z is a normalizing constant, and a and z are constrained so $\sum_{i=0}^{m} \text{cnt}(e_i) = x$ and $\min(\text{cnt}(e_i)) = 1$. We generated the specified number of EAs for each edge, randomly order all EAs, and assigned each a generation time as modeled by a homogeneous Poisson process with intensity λ. This was

[8] https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.degree_seq.configuration_model.html

TABLE II: Results against known-answer datasets.

Name	First EA	Last EA	Matched to	AMI Mean	AMI Stdev
Oracle	0	49,999	G_0	0.948	0.009
Oracle	50,000	99,999	G_1	0.708	0.044
EASEE	0	50,271.4	G_0	0.947	0.008
EASEE	50,272.4	99,999	G_1	0.704	0.038
Fixed Split 0.2	10,000	59,999	G_0	0.775	0.027
Fixed Split 0.4	20,000	69,999	G_0	0.096	0.011
Fixed Split 0.6	30,000	79,999	G_1	0.029	0.003
Fixed Split 0.8	40,000	89,999	G_1	0.094	0.014

repeated for each edge set. We concatenated the EA stream from $\mathcal{E}_1$ to the end of $\mathcal{E}_0$'s stream and increased the times for $\mathcal{E}_1$ so they continue right after the end of $\mathcal{E}_0$.

Community detection and comparison to fixed-width techniques: For the tests that follow, we used 10 stochastic blocks, $t_0 = 50,000$, $\lambda = 25$, $n = 2,000$, $p_{\text{in}_0} = 0.05$, $p_{\text{in}_1} = 0.04$, and $p_{\text{out}} = 0.005$. We generated 20 different EA streams with these parameters. For the community detection tests, we used Louvain community detection [26]. To compute community detection accuracy, we used AMI between the SBM-assigned communities, and the communities discovered by Louvain on each run. We formed a graph from the first EASEE-identified split of EAs and compared to the SBM for G_0; and formed a graph from the remaining EAs and compared to the SBM for G_1.

We compared the results from EASEE to two different types of solutions: (1) An Oracle that demonstrates the results from perfect accuracy in the split. For the Oracle, we formed a graph from the first 50,000 EAs and compared to the SBM for G_0, and formed a second graph from the second 50,000 EAs and compared to the SBM for G_1. (2) A variety of fixed-size datasets simulate a fixed-interval graph that correctly uses 50,000 EAs for the sample size, but does not monitor incoming EAs to identify change moments. This is a best-case scenario for several previous works that identify optimal fixed-size snapshots, but do not monitor for underlying change moments (e.g., [1] and [2]). The Fixed Split 0.X datasets simulate these with varying errors for when they split relative to the data change. For each of these, we formed 50,000-edge graphs from EA c to $c + 49,999$ for $c = 10,000$, $20,000$, $30,000$, and $40,000$. These graphs have 20%, 40%, 60% and 80% respectively of the EAs from G_1 mixed with parts of G_0. We compared these graphs' Louvain results with both G_0's SBM and G_1's SBM and (generously) report the best AMI.

Table II shows the results of our test. EASEE's runs are the only runs that are not precisely 50,000 EAs long, but its error in identifying the transition averages only 272.4 EAs late (stdev = 77.7; max = 448). That is, EASEE rapidly identified the shift in this data.

Although EASEE was a few hundred EAs later than perfect to identify the data shift, its AMI was nearly identical to the Oracle with a perfect split. This high AMI is particularly impressive when compared with how the naïve, fixed-interval approach did when it crossed the change moment. Even when the split contained only 10,000 of the later EAs, the AMI fell drastically. Furthermore, although G_1 had identifiable communities (both the Oracle and EASEE obtain AMIs around 0.7), its community structure was weak enough that when it traded 20% of its EAs for G_0 EAs (No Split 0.8), Louvain was unable to identify its communities well. When the EA mix was more even, community detection did even worse.

Thus, missing these change events – and forming graphs with components before and after the change – can drastically affect downstream tasks. EASEE is a critical tool to improve analytics for graphs from communication streams.

C. Sensitivity of EASEE to Initial EA Position

Now that we have identified (1) that EASEE correctly identifies change moments while ADAGE does not, and (2) that correctly identifying change moments is critical to downstream applications (while fixed-interval-size techniques will miss these and fail, except by random chance), we turn to analyzing EASEE's results on our real-world datasets.

Our first analysis tests how robustly EASEE responds to a standard real-world problem: When starting to form snapshots from a stream of EAs, there is no way to know if you are starting at a "good point" in the EA stream.

We simulated this problem by comparing EASEE results against 41 different start times on our real-world data. In an ideal setting, EASEE would recover the same snapshots started at any delay – as it would had it been applied to the full stream (for the shared observed portions). Slightly less ideally, EASEE would return to the snapshots predicted by the full stream after some brief period of instability.

We performed the following test on each of our datasets: (1) We omitted a fixed number (τ) of EAs at the start of the stream – providing EASEE the remaining data. (2) We reported the EA indices (relative to the full stream) where EASEE created snapshots. (3) We performed the above steps for $\tau = 0$; 5,000; 10,000; . . . ; 200,000. We stopped after 200,000 because (a) we needed many resulting snapshots to measure similarity across all our datasets, and (b) because our results showed strong convergence across all 41 different runs on all 7 datasets. (4) We compared the stability of the EA snapshot indices across different τs.

Fig. 12 shows EASEE run in this manner against the Enron dataset. Each column shows a different run against the Enron EA stream dropping the first τ EAs (horizontal axis). The dots indicate the EA when EASEE created a snapshot (vertical axis). Visual similarity is reached quickly.

Beyond the visual comparison in Fig. 12, we created a numerical comparison that takes these results from each of our seven datasets, grouped the last snapshots in each column, the second-to-last, until the first in each column. (Note that the first groups had fewer snapshots – some of the EASEE runs had not started yet.) We measured the standard deviation of the snapshot-ending EA indices in each group.

Fig. 13 shows these results. The horizontal axis is ordered by the snapshots generated and grouped as just described: Lower indices are the first snapshots taken. This shows that while EASEE does not identify the exact same snapshot-ending EA

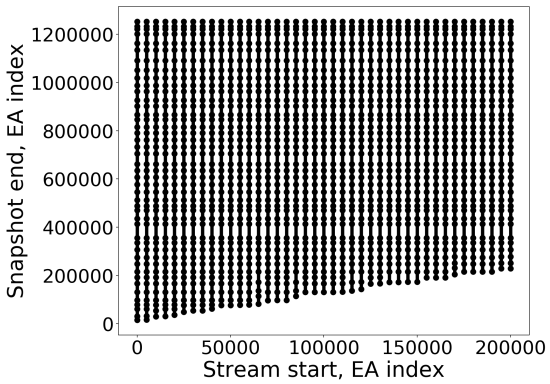


Fig. 12: EASEE sensitivity to different start times on the Enron dataset. Each column indicates a run of EASEE with a different starting delay (horizontal axis). Dots indicate EASEE snapshot times (vertical axis). Visual consistency is rapidly reached between one run and its immediate predecessor.

point for the first few snapshots after starting (compared to EASEE runs started on smaller τ), this initialization period quickly converges for all datasets. EU Emails and GameX Combat are the only two datasets where EASEE doesn't fully converge: As there are fewer EAs in these datasets, there are insufficient snapshots for convergence. Note that the near-zero values on most of these plots are actually precisely zero.

Thus, EASEE is very robust to differing start times in data streams. Once an early series of snapshots complete, EASEE will consistently give the same snapshots as it would have if started at a much earlier time. Furthermore, this indicates that EASEE is finding many consistent change points in these various datastreams.

D. Comparing and Merging Sufficient Snapshots

Thus far, our focus has been on analyzing EASEE's ability to identify minimal snapshots. In this section, we present many of the results of those snapshots and their associated cosine-similarity merge scores.

Although EASEE yields a set of merged non-overlapping snapshots, comparing all sufficient snapshots can provide insight into dataset structure. Thus, we show plots like Fig. 14 (for EU Emails), which we call "sufficient-snapshot similarity" (S^3) plots. The main diagonal has cosine similarity 1.0 as each snapshot is compared to itself. Off-diagonal elements $r \neq s$ are the cosine similarity of the r-th and s-th sufficient snapshots. Sufficient snapshots are ordered chronologically.

Fig. 14 shows the EU Emails dataset's 8 sufficient snapshots' similarities. All had similar node advertising rates, slightly dropping off with time. Edge similarity shows groups of similar snapshots: the first three, the middle three, and the last two with a threshold of $t_c = 0.7$. There is less similarity outside these groups.

In all datasets, node similarity is higher than edge similarity. To preserve space, we now show only edge similarities.

Fig. 15 shows the S^3 plot for the Enron Email dataset. We set $t_c = 0.5$. EASEE identifies sub-boxes that are above the threshold and merges them. We omit the EASEE merge boxes for all future plots for plot legibility.

Fig. 15 shows several key structures: First, there are periods of high neighbor similarity shown by boxes of lighter color down the main diagonal. Second, there are periods of slower variation, where snapshots are quite similar to several nearby sufficient snapshots, but slowly become less similar to those farther away (around snapshots $5-15$). Third, there are some periods of high neighbor similarity that repeat after periods of low similarity, as evidenced by the boxes of high similarity off the main diagonal.

This third pattern is instructive when considered in the real-world Enron context. The last high similarity box came immediately after Enron laid off 4,000 employees.[9] Although these snapshots come after Enron was drastically smaller, there are three periods before the layoff (with far more staff) with high similarity to this period (see arrows from the right). Thus, a significant proportion of the email characteristics of the post-layoff Enron had occurred previously. If we assume that the staff remaining at Enron after the layoffs were managers and lawyers tasked to handle urgent problems, these off-axis patterns lead to a question: What events happened in April and October 2001 that caused these same set of "handlers" to be so active then as well?

Fig. 16(a) shows the S^3 plot for GameX Messages. Like Enron, some periods of extended similarity repeat with other non-neighboring periods of extended similarity. These are split either by periods with low similarity even with their own neighbors, or periods that show some level of internal similarity.

Figs. 16(b) and (c) show two portions of the LANL Netflow S^3 plot. These have a shifted color scheme from all previous figures because many intervals are so similar that most of the plot would have been white. Fig. 16(b) shows the first 26 hours of sufficient snapshots. There appears to be a repeating network scan around every 70 minutes that drastically changes EA activity. This shows up as repeated dark blue bands. Ignoring those, daytime hours (0–7, 22-26) have lower internal similarity than nighttime hours (8–21).

Zooming out with Fig. 16(c), we see about 10 days' activity. The LANL dataset's timestamps were altered to remove specific dates and time of day. However, by looking at the daily activity patterns – where nights are similar to other nights – and combining the two-day period of night-like behavior, we can identify the likely weekend in this dataset. The Friday data is a lighter color than other weekdays (more similar to weekends), which may indicate LANL's flexible Friday work schedules.

Fig. 17 shows the S^3 plot for Stack Overflow. It shows some periods of reasonable similarity within sets of neighboring nodes, but no crisply defined squares of similarity. Some neighboring sufficient snapshots merge with some of their neighbors for low enough t_c, but there were no off-axis similarity blocks.

E. Warnings of building static graphs

A danger in building static snapshots from streaming communication data is that considerable data is lost. However,

[9] 12/2/01 https://www.theguardian.com/business/2006/jan/30/corporatefraud.enron

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3223614

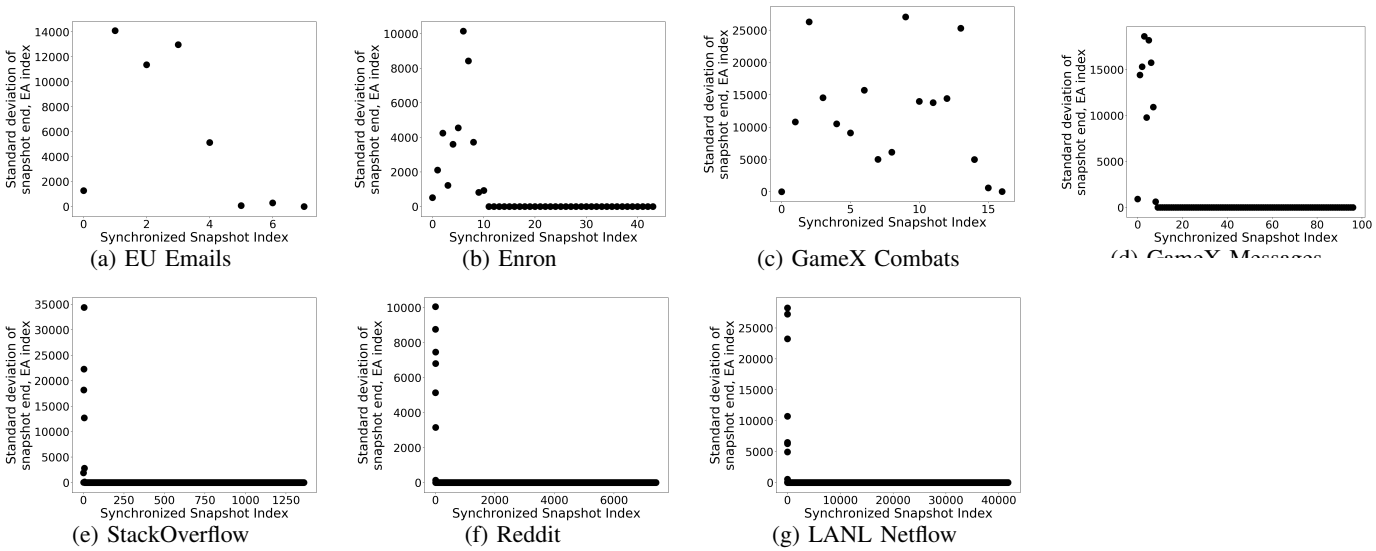


Fig. 13: Sensitivity analysis of EASEE against delay in the start time of the stream observations. We see that after a small number of snapshots, the downstream EASEE output is unaffected by different start times.

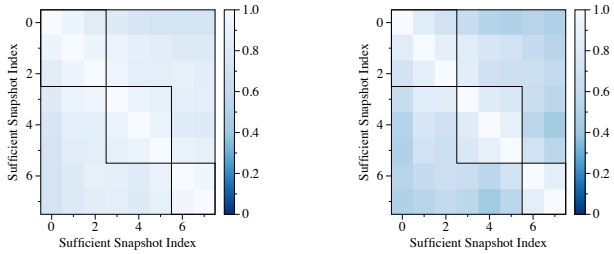


Fig. 14: Sufficient snapshot similarity for the EU Emails dataset. Node similarity is nearly uniformly large, but edge similarity breaks into three distinct subgroups.

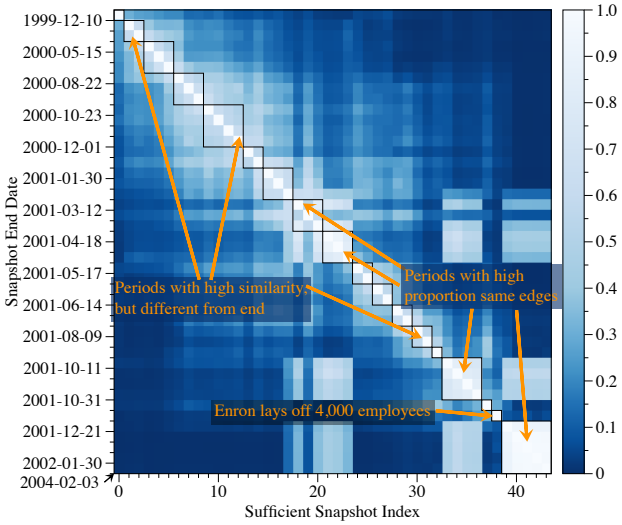


Fig. 15: Sufficient snapshot edge similarity for Enron Emails. There are several distinct periods shown. Further, while some of these periods are largely unique (arrows from middle-left), some periods repeat (arrows from top-right).

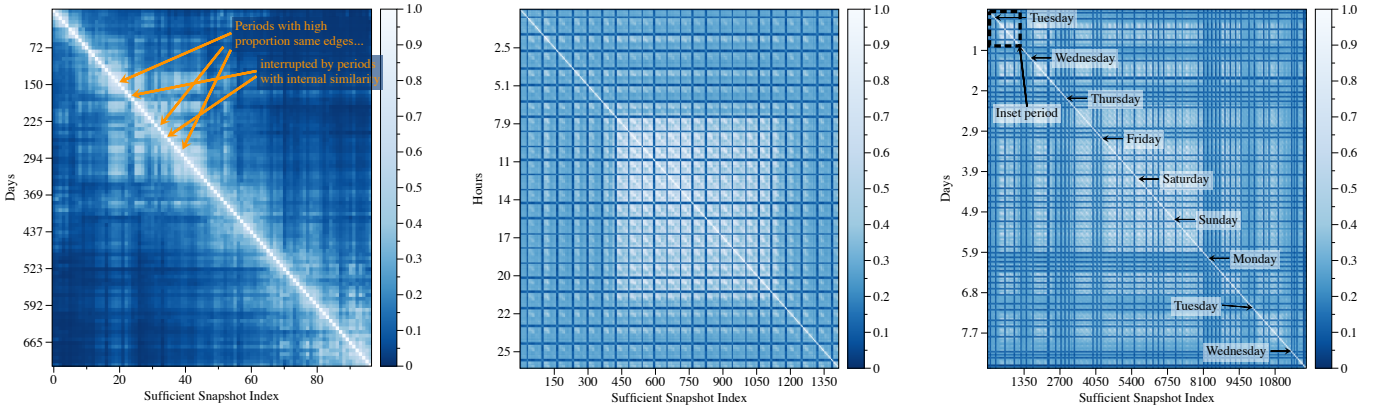
building static snapshots is desirable because of the many existing static graph analytics. Neighboring snapshots' cosine similarity indicates if the underlying data is shifting too quickly to form representative static snapshots. To our knowledge, EASEE is the only technique that can identify such change.

Sufficient snapshots identify minimal interval sizes that generate good graph representations. Neighboring sufficient snapshots aren't merged only when critical shifts occur in the underlying data. However, there are datasets where all neighboring sufficient snapshots show critical changes. That is, no neighboring intervals merge for a reasonable threshold.

Specifically, the GameX Combat and Reddit datasets show almost no between-interval similarities. The GameX Combat dataset split into 15 sufficient intervals, and none of the 14 neighboring pairs of intervals had a cosine edge similarity greater than 0.1. With 3 orders of magnitude more EAs, the Reddit dataset split into just over 7.4K sufficient intervals, but 92% of the neighboring pairs of intervals had cosine similarity less than 0.2, and less than 1% had cosine similarity greater than 0.5. Non-neighboring comparisons showed even less similarity. Both datasets showed higher node similarity across neighboring snapshots, indicating that while similar people often participated, edges were forming and dying between them within single sufficient intervals. Although only for a portion of the Reddit data, the scree plots for both of these datasets show this problem (Fig. 7).

This result indicates that within-snapshot changes were almost certainly happening as well. We would need much shorter snapshots to remove this within-snapshot change. However, EASEE already seeks minimal sufficient intervals to avoid undersampling. Thus, when all neighboring snapshots have very low similarity, this likely indicates that there is no safe trade-off between under- and oversampling: Static snapshots cannot reasonably represent the underlying data. We recommend testing for this case, e.g. for when most similarities are below a given threshold.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3223614



(a) GameX Messages (b) LANL First Day (c) LANL First 10 Days

Fig. 16: Sufficient snapshot edge similarity for (a) the GameX Messages and (b, c) parts of the LANL Network datasets. Both show considerable periods of similarity in both neighboring and separated intervals. The LANL dataset has a shifted color scheme because of high similarity.

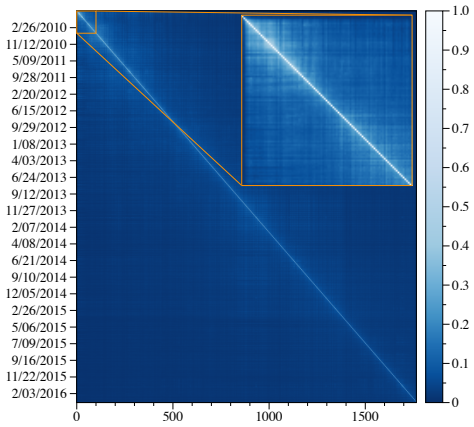


Fig. 17: Sufficient snapshot edge similarity for StackOverflow. Although the full image appears to have low similarity values, the inset in the upper right (just over 100 snapshots) shows that there is considerable similarity for neighboring snapshots.

Summary: In this section, we demonstrated the following:

- Using a synthetic dataset with a known change point, and a variety of rates and change times, we showed that EASEE is able to rapidly identify the changes if they are large, or smaller changes if they don't occur too close to EASEE initialization. It can't find changes that are too close to initialization and don't change as much data. ADAGE does not identify these change points at all.
- We demonstrated that identifying these changes is critical to accurate community detection. EASEE provided similar community detection scores (AMI) to an oracle, while correct-sized blocks that miss the change point do not.
- We showed that EASEE rapidly identifies exactly the same split indices after few sufficient snapshots even when initialized at very different points in the EA stream.
- We provide a visualization of merging results for EASEE against most of our datasets.
- We discuss how EASEE identifies that two of the datasets should not be used for static snapshot analysis.

VII. Densification

Now that we have demonstrated EASEE's accuracy and robustness, and shown several anecdotal results from EASEE's merge analysis, we use EASEE for down-stream analysis. Leskovec, et al. analyzed a variety of graph types that change over time, including two communication graphs (emails and autonomous systems) [13]. They used either cumulative interval sizes, or fixed-size sliding intervals. In their email dataset, Leskovec, et al. considered only edges that reciprocated (A talks to B, B responds at some point to A). While that does change the resulting graph (effectively eliminating all single-occurrence EAs and many low-occurrence EAs that were always the same direction), we include all data in our analysis. They found that as graphs age, they increase in density, and decrease in diameter. We now further that analysis for graphs from communication data by using EASEE to identify graph snapshots compared to the sliding interval graphs used by Leskovec, et al. As Reddit and GameX Combat were found to be problematic when forming static snapshots, we do not include them in this analysis.

Our technique largely matches Leskovec, et al.'s: We began by analyzing densification directly as they did; later, we analyzed density change against real-world time. Given a series of graph snapshots, we counted the number of unique nodes and unique edges in each snapshot, and plotted nodes against edges. We fitted the model

$$\text{unique edges} \propto (\text{unique nodes})^{\alpha} \tag{17}$$

to the networks, and studied the exponent α. For a given network, if the exponent α is larger than 1, the network is said to be *densifying*. We computed α by performing a linear regression (or a robustified linear regression) on the log of both edge and node counts. We compared these results for fixed-interval sliding intervals, a cumulative interval (growing from the start

of data to a given time), and EASEE's sufficient snapshots (output from Alg. 1). We used sufficient snapshots as their interval sizes are more consistent than merged interval sizes (which can vary wildly with the underlying data's consistency). This was the more fair comparison to fixed-size results. For the sliding intervals, we use overlapping, rolling periods of fixed temporal duration (as specified in results) – moving forward a single EA for each new snapshot.

Network	Interval Size	α	α, robust regression	Coefficient of Variation of $\frac{\text{Edges}}{\text{Nodes}}$
EU E-Mails	EASEE	1.271	1.335	0.047
EU E-Mails	2 months	2.484	2.84	0.093
EU E-Mails	4 months	1.449	1.449	0.025
EU E-Mails	8 months	1.159	1.125	0.015
EU E-Mails	Cumulative	4.082	4.642	0.315
Enron	EASEE	1.081	1.028	0.109
Enron	2 months	1.138	1.124	0.106
Enron	4 months	1.124	1.07	0.093
Enron	8 months	1.098	1.001	0.073
Enron	Cumulative	1.170	1.123	0.112
GameX Messages	EASEE	1.767	1.782	0.124
GameX Messages	2 months	1.803	1.687	0.110
GameX Messages	4 months	1.594	1.622	0.094
GameX Messages	8 months	–0.052	–0.259	0.090
GameX Messages	Cumulative	1.654	1.586	0.260
StackOverflow	EASEE	0.926	1.083	0.328
StackOverflow	2 months	0.646	0.642	0.247
StackOverflow	4 months	0.646	0.642	0.267
StackOverflow	8 months	0.64	0.639	0.263
StackOverflow	Cumulative	0.949	0.943	0.081
LANL	EASEE	1.447	1.441	0.173
LANL	1 second	1.055	1.082	0.154
LANL	1 minute	1.34	1.35	0.121
LANL	1 hour	0.274	0.295	0.284
LANL	1 day	0.203	0.199	0.285
LANL	3.5 days	11.432	10.615	0.073
LANL	1 week	5.501	1.109	0.059
LANL	Cumulative	0.849	0.886	0.197

TABLE III: Densification Experiment Results

Table III shows our results for the exponent α. For each network, the exponent can vary wildly depending how communication events are combined into snapshots. Enron's exponents are only slightly larger than 1. StackOverflow has exponents smaller than 1 independent of the temporal segmentation – although cumulative and EASEE both get close to 1. Oddly, LANL exhibits wildly varying change in exponent – approximately 1 at very small intervals; much smaller than 1 for medium intervals; and much larger than 1 for very large intervals. Thus, statements about the densification of a network are extremely sensitive to dataset and interval size.

One feature we saw in the plots that led to Table III's results is that in most datasets, more dense snapshots were not built from later data. That is, the latest snapshots were not the densest snapshots in any but the cumulative intervals. As the EASEE model has shown, densifying EAs (N0-Type) are the most common new-edge type when you don't take any snapshots (see Fig. 5). Thus, cumulative snapshots becoming denser with larger cumulative intervals is not a particularly surprising result. However, when snapshots are taken, later snapshots are not necessarily more dense than earlier snapshots.

To demonstrate this result further, we show a complementary view of density. Specifically, we studied the ratio edges/nodes for EASEE snapshots in order. In Table III, we report this quantity's coefficient of variation: It is generally low. That is, the density as measured by the ratio of unique edges to unique nodes does not exhibit a significant range. Fig. 18 shows these results for EASEE on each dataset. GameX Messages is the only dataset with even a minor increase in density (vertical axis) with time (horizontal axis). Enron and LANL are largely flat; the density of StackOverflow drops considerably at the beginning and then slowly decreases with time.

Lest any argue that these results are specific to EASEE, we present fixed-interval and cumulative results for the StackOverflow network in Fig. 19. Although the cumulative case has a slope of around 1 in the standard equation used by Leskovec (Eq. 17), the cumulative graphs increase in density drastically at first, but then decrease in density. This was the only case where the cumulative interval did not always increase in density. Furthermore, for all fixed-width intervals, the density largely decreases over time.

VIII. Discussion and Conclusion

EASEE provides a variety of insights into graphs built from communication event streams.

Densification: Leskovec, et al. [13] found that graphs that grow over time increase in density and shrink in diameter as the interval increases. Although our work studies only a subclass of their data,[10] our results support and add caveats to Leskovec's results.

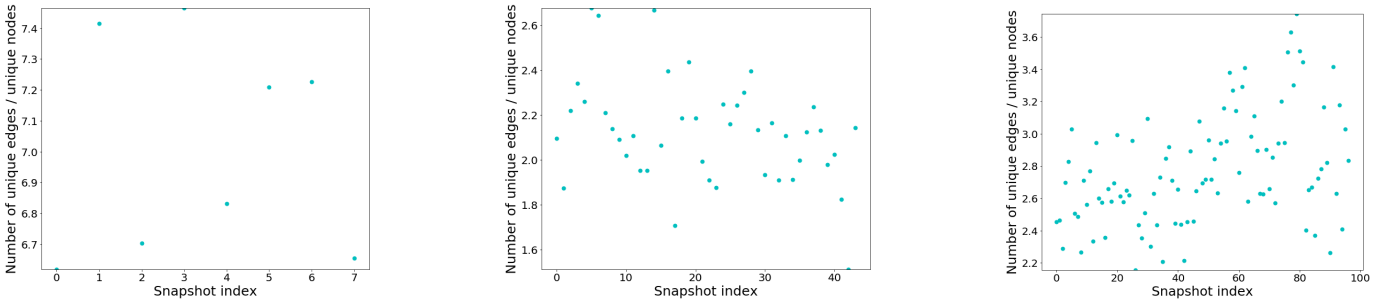
First, EASEE explicitly monitors EAs that cause densification (N0-type) and those that could increase diameter (N1-type and N2-type). As shown in Fig. 5, as more EAs arrive, N1- and N2-types are generally a very small fraction, while N0-type EAs often remain a non-trivial proportion. (Although absolute values vary, the trend of more N0-type than N1- or N2-type is generally similar for all our datasets.) Thus, densification generally increases as the snapshot size increases. Furthermore, as paths are added internally (N0-type) at a much higher rate than possible outside edges (N1-, N2-types), the diameter should decrease. Each internal edge added decreases path lengths for some of the nodes, eventually including the longest path between any two nodes.

However, if the interval responds to the underlying communication data, densification does not generally increase with later communication data (see Fig. 18).

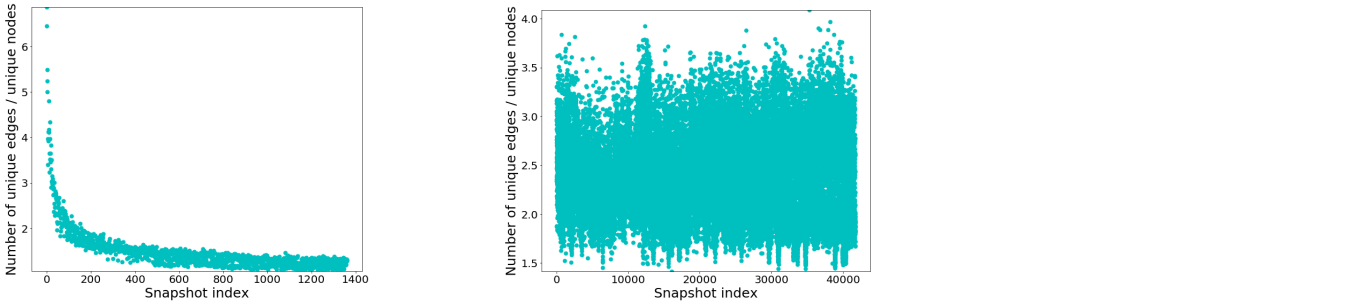
Second, EASEE's merge results show that often intervals *should* end. As the community-detection analysis shows, neighboring periods can be so different that merging them confounds later analyses – the resulting graph contains many paths that never were active together. Thus, some of Leskovec et al.'s results require the caveat that there are likely multiple distinct static graphs within their larger-interval graphs. In fact, the sufficient snapshots and fixed-interval snapshots for StackOverflow have a clear decrease in density over time (Figs. 18(d) and 19).

Sudden changes: Our results show that EASEE can rapidly identify sudden changes in datastreams with low latency (Fig. 10 and Table II). Such changes occur regularly in many datasets (Figs. 15 and 16). Detecting these changes rapidly and

[10]EASEE requires repeat edges as in most communication systems; Leskovec et al.'s analysis includes citation networks where edges never repeat.

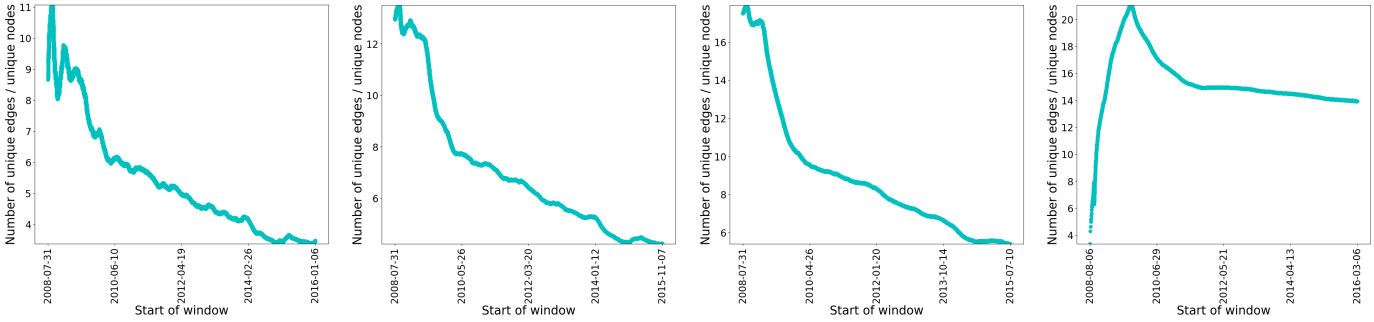


(a) EU Emails (b) Enron (c) GameX Messages



(d) StackOverflow (e) LANL Netflow

Fig. 18: For each dataset and the EASEE algorithm, we plot the ratio of unique edges to unique nodes in the snapshots in time order. Note that density does not generally increase with time.



(a) 2-Month Rolling Interval (b) 4-Month Rolling Interval (c) 8-Month Rolling Interval (d) Cumulative Interval

Fig. 19: Densification results for the StackOverflow dataset on rolling intervals of size 2, 4, and 8 months, and cumulative intervals. Note that there is no evidence for increasing density with time.

creating new graph snapshots is a critical step in generating static graphs from communication streams. However, previous research does not handle these results: ADAGE failed to detect any change moment directly (Sec. VI-A); fixed-width techniques will fail as often as not which negatively affects down-stream applications (Sec. VI-B).

Graph stability: Our results show that some communication streams have continual underlying catastrophic shifts (Sec. VI-E). Creating static snapshots from such data should be considered dangerous. These results argue strongly for temporal network analysis on such datasets, but great caution when considering static graph analysis.

Furthermore, given a few snapshots to converge, EASEE identifies exactly the same snapshots whether it was started recently or many EAs earlier (Sec. VI-C).

Repeating periods: Our results show that some communication streams show non-neighboring periods with repeating behavior (Figs. 15 and 16(a) and (c)). Such periods may be of considerable interest to analysts studying the underlying dataset: What was happening at Enron in April 2001, October 2001, and January 2002 that led to such high similarity? Furthermore, should an off-line merging step consider merging such snapshots?

Weighted edges: We tested neighboring sufficient snapshots with various static graph similarity measures. We found that neighboring unweighted snapshots were quite distinct – even when they had high cosine similarity. This is likely because unweighted graphs consider edges with only one advertisement of equal weight to edges with dozens or more. As our cosine similarity results show, these neighboring periods can still have high weighted similarity. Thus, we recommend weighting the edges of the final static snapshots with the number of EAs and using weighted analyses for down-stream analyses.

Dying edges: We are aware of no technique that directly

detects dying edges within a single snapshot. EASEE also cannot explicitly detect dying edges as they are only shown by their lack: There are no "edge dead" advertisements in communications data. EASEE handles this issue by limiting sufficient intervals to a minimal period until convergence, and then comparing the cosine similarity of neighboring snapshots to decide to merge them. If an edge seen in one snapshot does not advertise in the next, then it effectively "dies."

We tried modeling the probability distribution of EA inter-arrival times and identifying dead edges as those which had not advertised in a statistically unlikely period since their last EA. However, this solution is quite complicated due to the following issues: First, the EA inter-arrivals best followed long-tailed distributions like Weibull or Pareto distributions. This leads to very long periods before expecting to see a new EA with non-trivial probability. Second, as indicated in Fig. 4, different edges advertise at wildly differing rates. Thus, it would have required different distributions and monitors for each edge. Third and finally, edges that advertise only once within an entire dataset may either have advertised exactly once ever, or may have advertised only once within the available data – sometimes several years long. Accurately representing such highly infrequent advertisers is extremely difficult.

Conclusion: We presented EASEE (Edge Advertisements into Snapshots using Evolving Expectations). EASEE runs in-line with streaming communication data and directly monitors EA types. Using EA types' evolving expectations, EASEE identifies minimal sufficient intervals where the snapshot has either converged or the underlying data has shifted. The merging step allows for larger snapshot intervals but explicitly monitors for oversampling. Furthermore, off-line analysis of EASEE's results can identify repeating patterns.

ACKNOWLEDGMENTS

The views expressed in this article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Wendt, Field, Phillips, Wilson, and Prasadan were supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. Soundarajan is supported by the U.S. Army Research Office grant number W911NF1810047. Bhowmick is supported by NSF Awards #1725566 and #1900765.

REFERENCES

[1] J. Sun, S. Papdimitriou, P. S. Yu, and C. Faloutsos, "GraphScope: Parameter-free mining of large time-evolving graphs," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[2] R. Sulo, T. Y. Berger-Wolf, and R. Grossman, "Meaningful selection of temporal resolution for dynamic networks," in *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, 2010.

[3] G. Krings, M. Karsai, S. Bernhardsson, V. D. Blondel, and J. Saramaki, "Effects of time window size and placement on the structure of an aggregated communication network," *EPJ Data Science*, 2012.

[4] R. S. Cáceres, "Temporal scale of dynamic networks," Ph.D. dissertation, University of Illinois at Chicago, 2013.

[5] S. Soundarajan, A. Tamersoy, E. Khalil, T. Eliassi-Rad, D. H. Chau, B. Gallagher, and K. A. Roundy, "Generating graph snapshots from streaming edge data," in *WWW '16 Companion: Proceedings of the 25th International Conference Companion on World Wide Web*, 2016, pp. 109–110.

[6] B. Fish and R. S. Cáceres, "A supervised approach to time scale detection in dynamic networks," arXiv:1702.07752v1, Tech. Rep., 2017.

[7] G. K. Orman, N. Türe, S. Balcisoy, and H. A. Boz, "Finding proper time intervals for dynamic network extraction," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, Mar. 2021.

[8] A. Lakhina, J. W. Byers, M. Crovella, and P. Xie, "Sampling biases in IP topology measurements," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2003.

[9] D. Achlioptas, A. Clauset, D. Kempe, and C. Moore, "On the bias of Traceroute sampling: or, power-law degree distributions in regular graphs," *Journal of the ACM*, vol. 56, no. 4, April 2005.

[10] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of ACM Knowledge Discovery and Data Mining (KDD)*, 2006.

[11] A. S. Maiya and T. Y. Berger-Wolf, "Sampling community structure," in *Proceedings of the International Conference on the World Wide Web (WWW)*, 2010.

[12] ——, "Benefits of bias: Towards better characterizations of network sampling," in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2011.

[13] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, 2007.

[14] L. E. C. Rocha, N. Masuda, and P. Holme, "Sampling of temporal networks: Methods and biases," *Physical Review E*, vol. 96, 2017.

[15] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, pp. 97–125, 2012.

[16] S. Ross, *A First Course in Probability*. New York: McMillan, 1994.

[17] J. H. van Lint and R. M. Wilson, *A Course in Combinatorics*. Cambridge, UK: Cambridge University Press, 2001.

[18] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in temporal networks," in *ACM International Conference on Web Search and Data Mining*, 2017.

[19] J. Hessel, C. Tan, and L. Lee, "Science, askscience, and badscience: On the coexistence of highly related communities," in *Proceedings of the AAAI Conference on Web and Social Media*, 2016.

[20] P. Liu, A. R. Benson, and M. Charikar, "Sampling methods for counting temporal motifs," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2019.

[21] M. J. M. Turcotte, A. D. Kent, and C. Hash, *Unified Host and Network Data Set*. World Scientific, 2018, ch. 1, pp. 1–22.

[22] S. Ioffe, "Improved consistent sampling, weighted minhash and l1 sketching," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 246–255.

[23] D. A. Jackson, "Stopping rules in principal components analysis: A comparison of heuristical and statistical approaches," *Ecology*, vol. 74, no. 8, pp. 2204–2214, 1993.

[24] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clustering comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[25] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[26] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, 2008.