Generating Graph Snapshots from Streaming Edge Data

Sucheta Soundarajan[*] Acar Tamersoy[†] Elias B. Khalil[†] Tina Eliassi-Rad[‡]
Duen Horng Chau[†] Brian Gallagher[§] Kevin Roundy[♯]

[*]Syracuse University [†]Georgia Institute of Technology [‡]Rutgers University
[§]Lawrence Livermore National Laboratory [♯]Symantec Inc.

[*]susounda@syracuse.edu [†]{tamersoy, lyes, polo}@gatech.edu
[‡]eliassi@cs.rutgers.edu [§]bgallagher@llnl.gov [♯]kevin_roundy@symantec.com

ABSTRACT

We study the problem of determining the proper aggregation granularity for a stream of time-stamped edges. Such streams are used to build time-evolving networks, which are subsequently used to study topics such as network growth. Currently, aggregation lengths are chosen arbitrarily, based on intuition or convenience. We describe *ADAGE*, which detects the appropriate aggregation intervals from streaming edges and outputs a sequence of *structurally mature* graphs. We demonstrate the value of ADAGE in *automatically* finding the appropriate aggregation intervals on edge streams for belief propagation to detect malicious files and machines.

General Terms

Algorithms, Design, Performance, Experimentation.

Keywords

Aggregating edge streams, time-evolving networks.

1. INTRODUCTION

We address the problem of determining *proper* intervals for aggregating a stream of time-stamped edges into a sequence of *structurally mature* networks. We define a structurally mature network as one that has stabilized with respect to a network statistic, such as the exponent of the degree distribution. Current literature in time-evolving networks[1] frequently selects an arbitrary fixed-length aggregation interval (e.g., one day,). This approach has three shortcomings. (1) If the interval is too short, the graph may lack sufficient structure for analysis (e.g., consider running belief propagation on a set of disconnected edges). (2) If the interval is too long, fine-grained changes get lost. (3) If the data streams at a variable rate, a fixed-length interval may not be appropriate. To address these issues, we introduce *ADAGE*, the *Adaptable Graph Edge Interval Method*, which *automatically* identifies variable-length aggregation intervals, each giving rise to a structurally mature graph. We report results on an edge stream from Symantec, for the task of malicious file/machine detection via belief propagation.

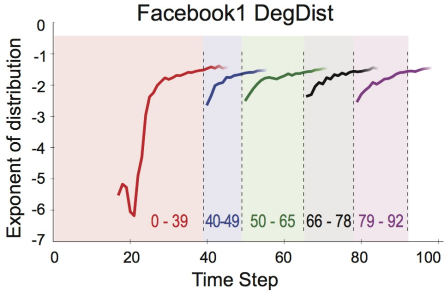

Figure 1: Dashed lines depict intervals found by ADAGE, using degree distribution exponent, on Facebook wall-postings. Each time step is one hour.

2. RELATED WORK

The literature on partitioning a data stream is vast. For example, Kiernan et al. [3] identify disjoint summaries of an event stream. Keogh et al. [2] present a solution for time series segmentation. The DAPPER algorithm [1] partitions a timeline of streaming network data into disjoint interval snapshots by examining edge persistence for change-points. Unlike ADAGE, DAPPER finds locations where the streaming data suddenly changes, as opposed to structurally mature snapshots, and requires the entire timeline upfront.

3. PROPOSED METHOD: ADAGE

ADAGE is an online method for aggregating streaming edges into a sequence of structurally mature networks. Given a network statistic (e.g., exponent of the degree distribution), ADAGE aggregates streaming edges into a network until the value of the statistic converges. Figure 1 depicts the time intervals detected by ADAGE on Facebook wall-postings vs. the exponent of the degree distribution of the composed graphs. In this graph, an edge represents that a user posted on another user's wall. Each time step is an hour. 40 hours worth of data were required to compose a graph with a stable degree distribution exponent. 10 hours were needed to generate the next structurally mature graph.

ADAGE takes as input a sequence of edge sets $E_1, E_2, \ldots$ arriving at times $T_1, T_2, \ldots$, and a function $f(G)$, which out-

[1] We use the terms graph and network interchangeably.

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2889398 .

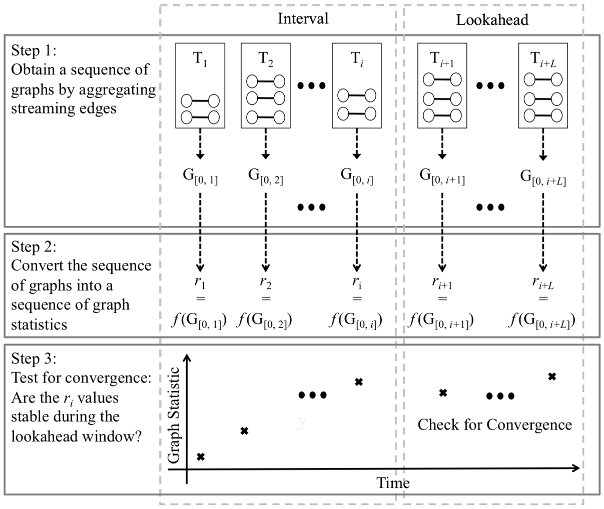
Interval
Lookahead
Step 1:
Obtain a sequence of graphs by aggregating streaming edges
T_1
T_2
T_i
T_{i+1}
T_{i+L}
$G_{[0,1]}$
$G_{[0,2]}$
$G_{[0,i]}$
$G_{[0,i+1]}$
$G_{[0,i+L]}$
Step 2:
Convert the sequence of graphs into a sequence of graph statistics
$r_1 = f(G_{[0,1]})$
$r_2 = f(G_{[0,2]})$
$r_i = f(G_{[0,i]})$
$r_{i+1} = f(G_{[0,i+1]})$
$r_{i+L} = f(G_{[0,i+L]})$
Step 3:
Test for convergence: Are the r_i values stable during the lookahead window?
Graph Statistic
Time
Check for Convergence

Figure 2: ADAGE **Overview: Edges are aggregated until convergence is detected on the chosen statistic.**

puts the value of a specified statistic on graph G. At each time T_i, f is applied to the current aggregated graph G_i to obtain value r_i. These values are inspected for convergence.

ADAGE can take any network statistic, such as exponent of degree distribution, number of nodes in the largest connected component, etc. The choice of statistic depends on the nature of phenomenon under study. For example, the exponent of the degree distribution is a good statistic if the phenomenon is expected to exhibit the Pareto principle.

Figure 2 provides an overview of ADAGE, which begins at time T_1 and aggregates data until the statistic converges. To determine whether convergence has occurred at time T_i, ADAGE examines the value r_i and the set of values $\{r_{i+1}, \ldots, r_k\}$ seen during the lookahead window $[T_{i+1}, T_{i+L}]$. The length L of the lookahead window is calculated using a parameter b, where $L = b \times i$ (L depends on the length of the interval so far).[2] The allowed deviation in values $r_i, \ldots, r_k$ is controlled by threshold parameter c: the difference between the largest and smallest values cannot exceed threshold t, which is equal to c times the smallest value (assuming all values are positive).[3][4] Once convergence is detected, ADAGE outputs the graph and restarts the aggregation process.

To avoid computing the statistic on every snapshot, ADAGE uses a binary search to choose which values to compare. Suppose ADAGE is considering time T_i, and so examines values $r_i, \ldots, r_k$. The allowed deviation of the statistic among these values is represented by the threshold $t = c \times \texttt{min}(\{r_i, \ldots, r_k\})$ (where c is the threshold parameter.) Let T_j be the midpoint between T_i and T_k. ADAGE calculates an estimate $\hat{t}$ on t, where $\hat{t}$ is equal to $c \times \texttt{min}(\{r_i, r_j, r_k\})$. If $\texttt{max}(\{r_i, r_j, r_k\}) - \texttt{min}(\{r_i, r_j, r_k\}) \leq \hat{t}$, then ADAGE calculates all intermediate values $r_i, \ldots, r_k$ to check for convergence. Otherwise, $\texttt{max}(\{r_i, r_j, r_k\}) - \texttt{min}(\{r_i, r_j, r_k\}) > \hat{t}$ and ADAGE cannot possibly have converged at time T_i. In this case, ADAGE recalculates a new estimate on t as follows. If $| r_j - r_k | > c \times \texttt{min}(\{r_j, r_k\})$, then convergence is not possible at any time between T_i and T_j; thus values $r_{i+1}, \ldots, r_{j-1}$ are not calculated. This process is repeated by looking at the midpoint value between r_j and r_k, and so on. In this way, ADAGE finds the earliest time when convergence might occur.

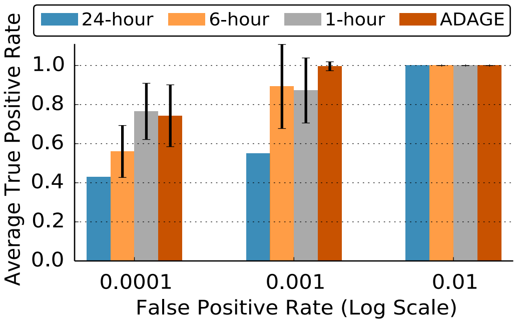

Figure 3: Detecting malicious files/machines via belief propagation on Symantec edge streams. ADAGE **intervals have the best performance.**

4. EVALUATION AND DISCUSSION

Evaluation of ADAGE is challenging, as we lack a ground truth. Here, we evaluate ADAGE on the task of detecting malicious files and machines via belief propagation. (Data provided by Symantec.) The dataset consists of an edge stream of `file i resides on machine j`.[5] We have a total of $3M$ edges connecting $0.6M$ nodes. Some files or machines are known to be malicious, and we use belief propagation to predict which others are likely to be malicious. We choose the exponent of the degree distribution as the statistic used by ADAGE, because we assume that mature `file`×`machine` graphs have power-law degree distributions. Figure 3 shows 10-fold cross-validation results of various aggregation methods. ADAGE *automatically* finds intervals that match or outperform fixed-length intervals. ADAGE took on average 4.19 seconds to detect each interval, each with ≈$0.5M$ edges.

By detecting convergence of a statistic of interest, ADAGE automatically identifies the proper aggregation granularity on streams of edges. Our future work includes being able to take prior knowledge such as periodicity into account.

5. ACKNOWLEDGEMENTS

This work was funded in part by LLNL under Contract DE-AC52-07NA27344, NSF CNS-1314603 and 1526254, and DTRA HDTRA1-10-1-0120.

6. REFERENCES

[1] R. S. Caceres. *Temporal Scale of Dynamic Networks.* PhD thesis, University of Illinois at Chicago, 2013.

[2] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.

[3] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. *TKDE*, 3(4):21:1–21:31, 2009.

[2] To avoid detecting convergence after very short intervals, we set the window length to be at least 10.

[3] We assume values are all positive or all negative.

[4] A parameter study suggested setting $b, c = 0.1$. Automatically selecting these parameters is part of our future work.

[5] www.symantec.com/about/profile/universityresearch/sharing.jsp