Sampling methods for counting temporal motifs

Paul Liu
Stanford University
Department of Computer Science

Austin Benson
Cornell University
Department of Computer Science

Moses Charikar
Stanford University
Department of Computer Science

MOTIVATION

Temporal network data is extremely common.

Private communication
e-mail, phone calls, text messages, instant messages

Public communication
Q&A forums, Facebook walls, Wikipedia edits

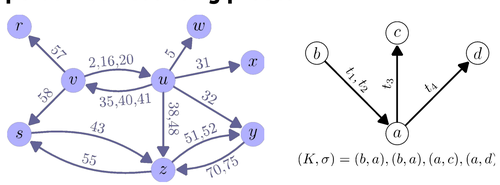
Payment systems
credit card transactions, cryptocurrencies, Venmo

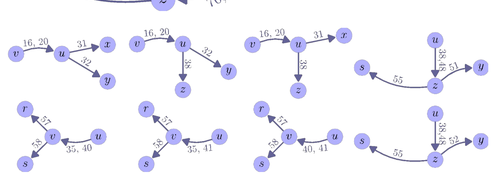
Technical infrastructure
packets over the Internet, messages over supercomputer

Temporal graphs and motifs are a way to abstract and analyze network data.

WHAT IS A TEMPORAL MOTIF?

[Paranjape-Benson-Leskovec 17] precisely defines the temporal motif counting problem.





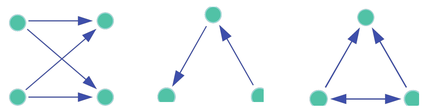
Temporal network motif..
- Directed multigraph with k edges
- Edge ordering
- Max. time span δ = 25.

Motif instance.
k temporal edges that match the pattern that all occur within δ time.

MOTIFS IN PRACTICE

Motifs, or small subgraph patterns, are commonly used to analyze static (non-temporal) networks.

1. Common feature for anomaly detection, role discovery, and other network machine learning problems.
   [Noble-Cook 03; Sun+ 07; Henderson+ 12; Rohe-Qin 13; Rossi-Ahmed 15; Benson-Gleich-Leskovec 16]



2. Finding fundamental components of complex systems.
   [Milo+ 02]
   a. Triangles in social networks.
      [Rapoport 53; Granovetter 73; Watts-Strogatz 98]
   b. Bi-directed length-2 paths in brain networks.
      [Sporns-Kötter 04; Sporns+ 07; Honey+ 07]

Existing methods for temporal analysis are insufficient.

- Prior algorithms focused on enumeration [Mackey+ 18] or exact counts for small motifs [Paranjape-Benson-Leskovec 17].
- Algorithms were extremely memory extensive.
- Compute times on the order of days for our largest datasets, and could not be done in a streaming manner.

How do we enable real-time motif analysis for high-throughput temporal network data?

Idea: Majority of applications only require *approximate* motif counts.

OUR CONTRIBUTIONS

Our sampling framework accelerates and parallelizes *existing* algorithms, yielding two orders of magnitude speedups.

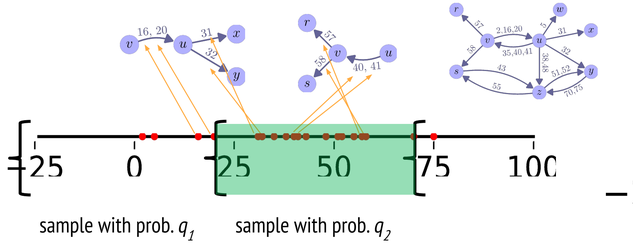
Additional contributions.
- Theoretical foundation for temporal motif counting, showing that it is NP-hard.
- A sampling framework to accelerate existing temporal motif counting algorithms.
- New sampling algorithms are memory-efficient, enables otherwise infeasible computations, and can be done in a streaming fashion.

		running time (seconds)			
dataset	# temporal edges	exact	sampling	parallel sampling	error
StackOverflow	47.9M	221.7	93.10	5.208	4.9%
EquinixChicago	345M	481.2	45.50	5.666	1.3%
RedditComments	636M	X	6739	2262	–

ALGORITHM OUTLINE

We find motifs in sampled windows and re-scale counts.

Sampling window length $w > \delta$ ($\delta = 25$, $w = 50$).
Choose random shift s uniformly from [0, 1, …, w - 1] (s = 20).



Partition, sample, compute.
- Partition data into the window they lie in.
- Sample some of the windows.
- Run exact motif counting algorithm on each sampled window.

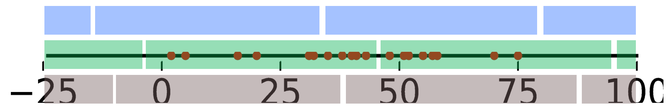
Non-trivial issues.
- How do we re-scale motif counts?.
- Motifs can cross sampling windows. How do we mitigate this?.
- How do we choose sampling probabilities qj?

ALGORITHM OUTLINE II

Our algorithm in a nutshell: partition, sample, compute.

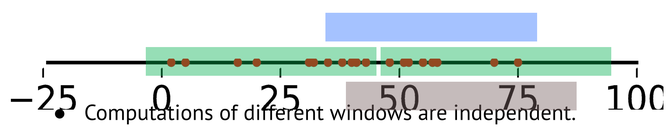
Partition data into windows	Parallel Importance Sampling	Count and upscale
Choose a random shift s, and partition data into windows offset by shift s.	For jth window, sample with probability qj.	Upscale counts of motif instances depending on their duration d(M).

Theorem. If we sample window j with prob. q_j, then upscaling each found motif instance by $(1 - d(M) / w) / q_j$ is an unbiased estimator, where $d(M)$ is the duration of the motif instance M.



- Using multiple random shifts and averaging the estimates reduces variance by capturing motifs that cross sampling intervals.
- s = 20, s = 32, s = 37
- Computation over each shift is parallelizable.

Computation over windows is naturally *streaming*.



- Computations of different windows are independent.
- Old data from longer than one window ago can be thrown away.
- Multiple estimators can be run in parallel to ensure accuracy.

INTERESTED IN MORE?

Slides. tinyurl.com/wsdm19
Paper. *arXiv:1810.00980*
Code. tinyurl.com/wsdm19-code
paul.liu@stanford.edu