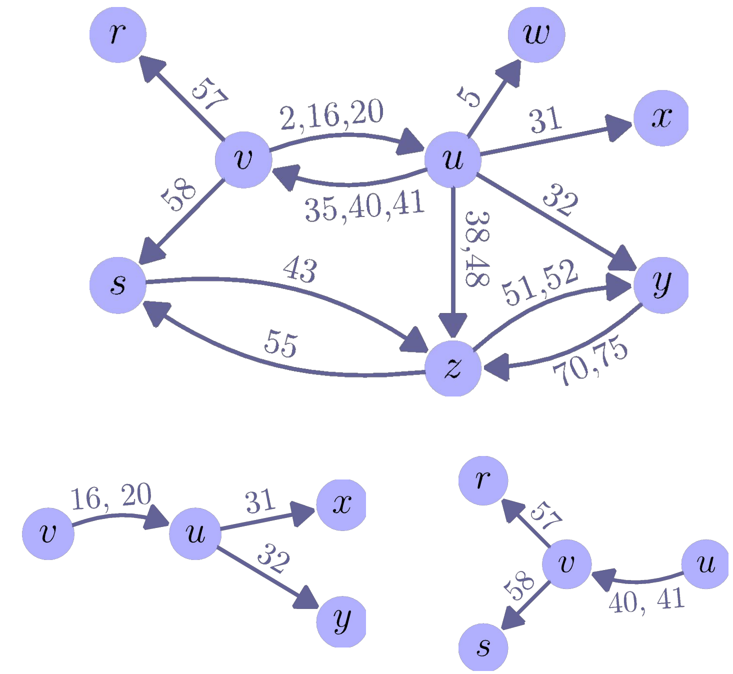
(based on slides by Austin Benson)

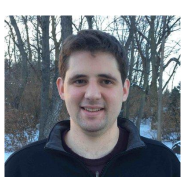
Sampling methods for counting temporal motifs



Paul Liu · Stanford University

Slides. `tinyurl.com/wsdm19`

Paper. *arXiv:1810.00980*




Joint work with
Austin Benson (Cornell) &
Moses Charikar (Stanford)

Stanford University

Temporal network data is extremely common.

Private communication

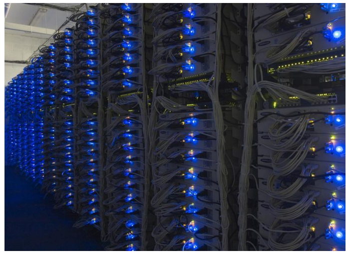
e-mail, phone calls, text messages, instant messages

Public communication

Q&A forums, Facebook walls, Wikipedia edits

Payment systems

credit card transactions, cryptocurrencies, Venmo

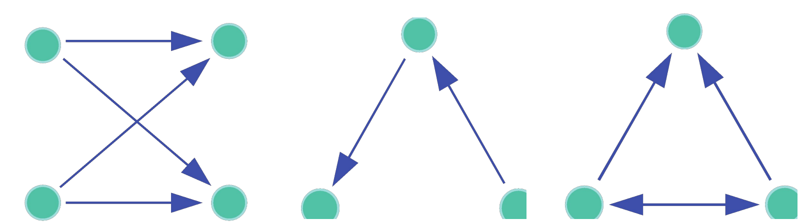


Technical infrastructure

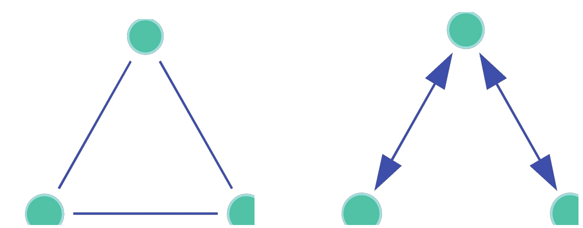
packets over the Internet, messages over supercomputer

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]

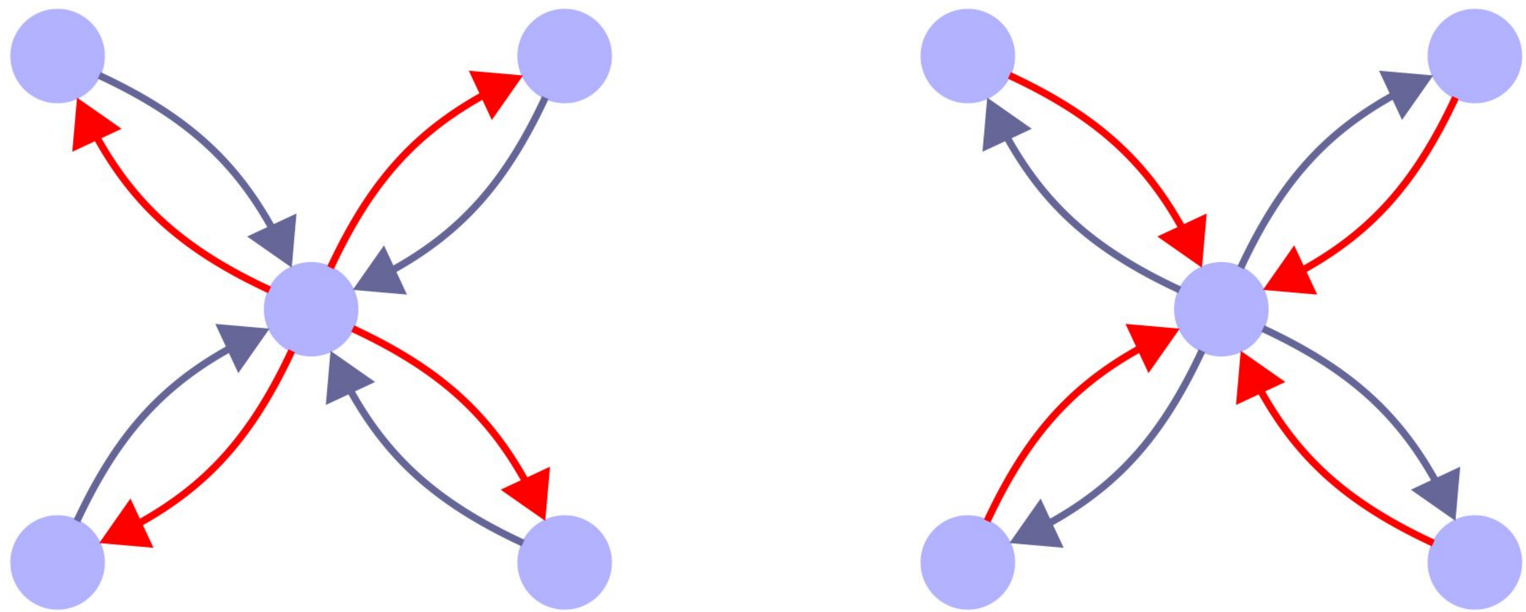


2. Finding fundamental components of complex systems. [Milo+ 02]
   - Triangles in social networks.
     [Rapoport 53; Granovetter 73; Watts-Strogatz 98]
   - Bi-directed length-2 paths in brain networks.
     [Sporns-Kötter 04; Sporns+ 07; Honey+ 07]



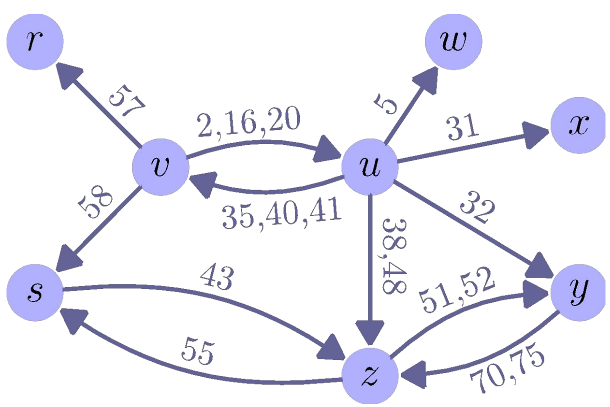
Temporality adds important context to motifs.

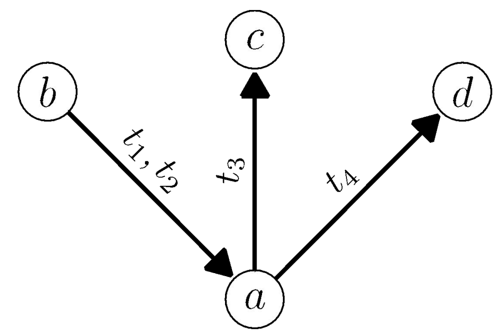
Red edges occur before **blue** edges.
What is the difference between the following *temporal* motifs?



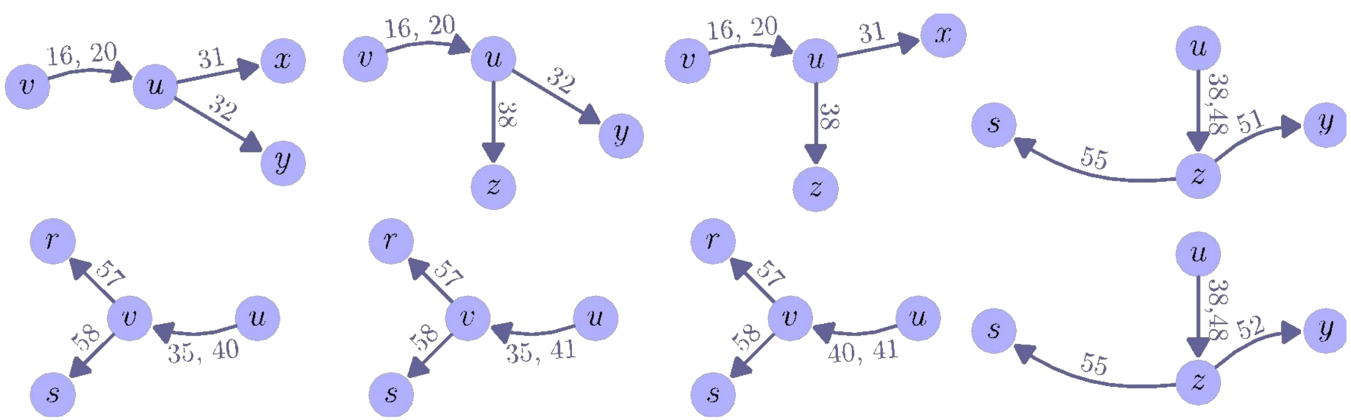
How does context change if δ = 1 day? 1 month? 1 year?

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





$(K, \sigma) = (b, a), (b, a), (a, c), (a, d)$



Temporal network motif.

1. Directed multigraph with *k* edges
2. Edge ordering
3. Max. time span δ = 25.

Motif instance.

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

There is a need to have scalable algorithms for real-time temporal motif analysis.

- 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?

Majority of applications only require *approximate* counts.

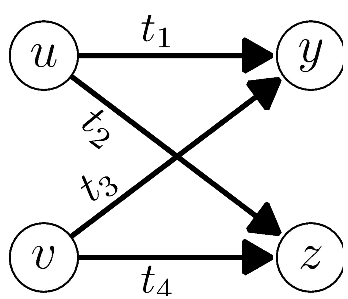
From exact to approximate: random sampling.

Idea. Use *importance sampling* to accelerate existing motif algorithms.

Our 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, and can be done in a streaming fashion.

Parallel sampling yields about two orders of magnitude speedup and enables otherwise infeasible computations.



Time scale δ = 1 day.
16 threads.

		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	–

Using backtracking algorithm from [Mackey+ 18] as a sub-routine.

From exact to approximate: random sampling.

“Easy” solution: sample subset of edges from graph and run exact algorithm on it.

How do we sample? (Two simple ideas)

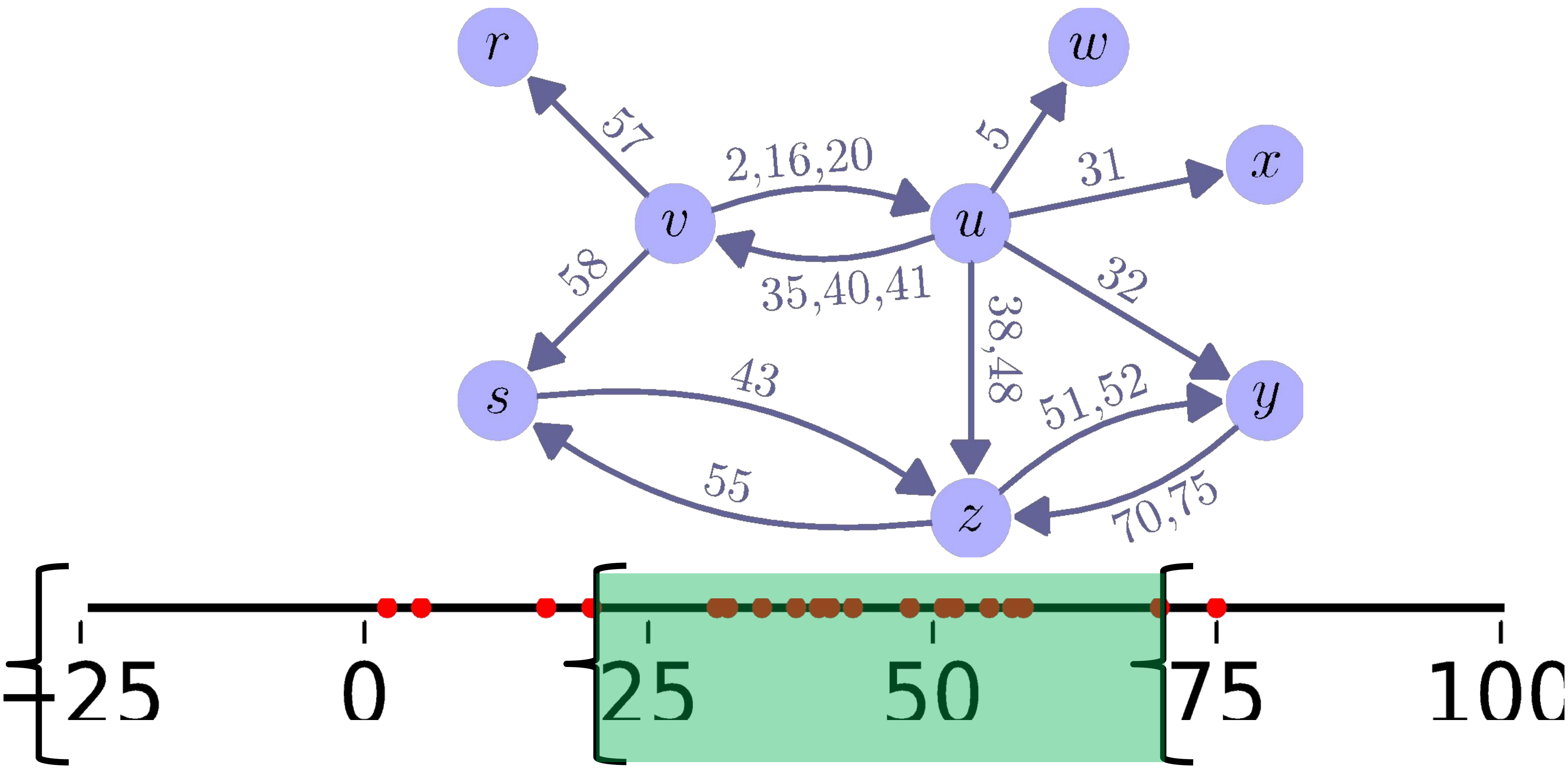
1. Sample uniformly random subset of edges? ✘ edges sampled often are too far apart in time.

2. Sample uniformly random windows of duration δ? ✘ contribution from windows are uneven. May miss temporally dense regions of edges.

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

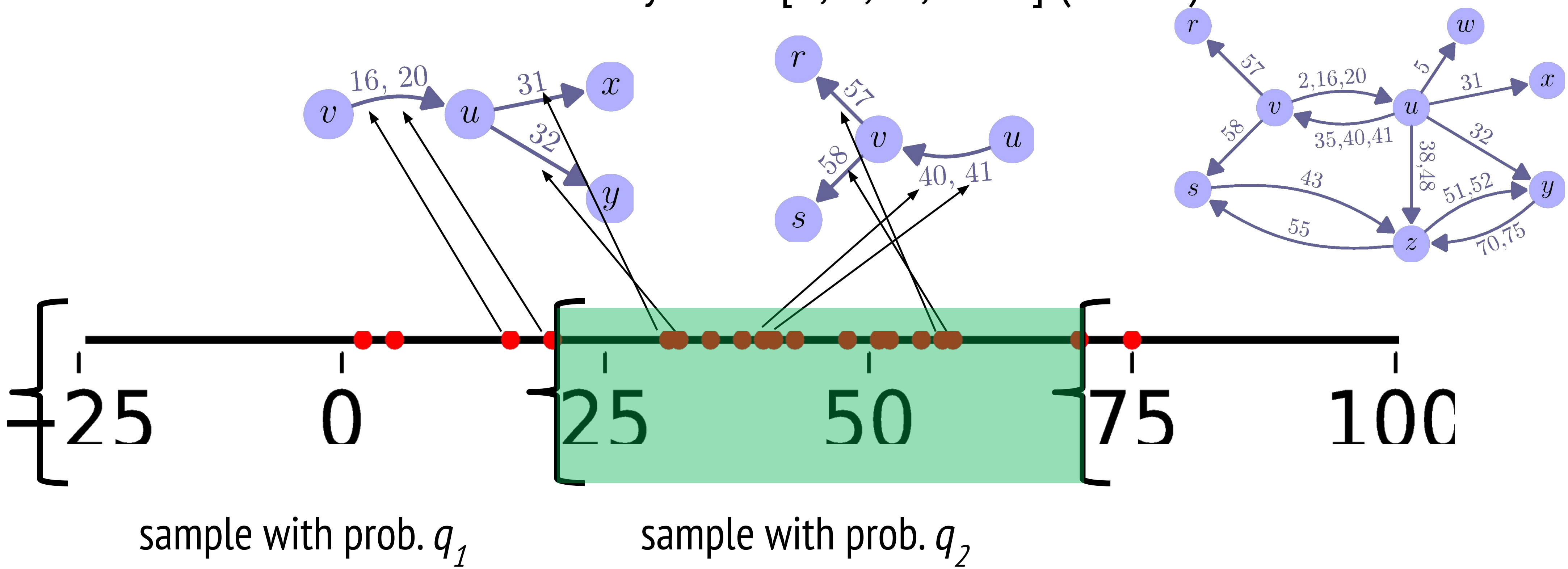
Sampling window length $w > \delta$ ($\delta = 25$, $w = 50$).

Choose random shift s uniformly from [0, 1, ..., w - 1] (s = 20).



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

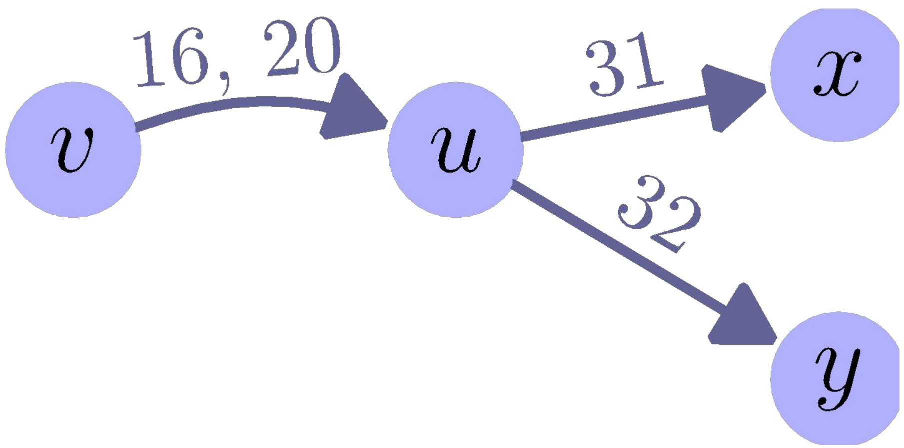
Sampling window length $w > \delta$ ($\delta = 25$, $w = 50$).
Choose random shift s uniformly from [0, 1, ..., w - 1] (s = 20).



1. How do we re-scale exact counts?
2. Motifs can cross sampling intervals. How do we mitigate this?
3. How do we choose sampling probabilities q?

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] ($s = 20$).



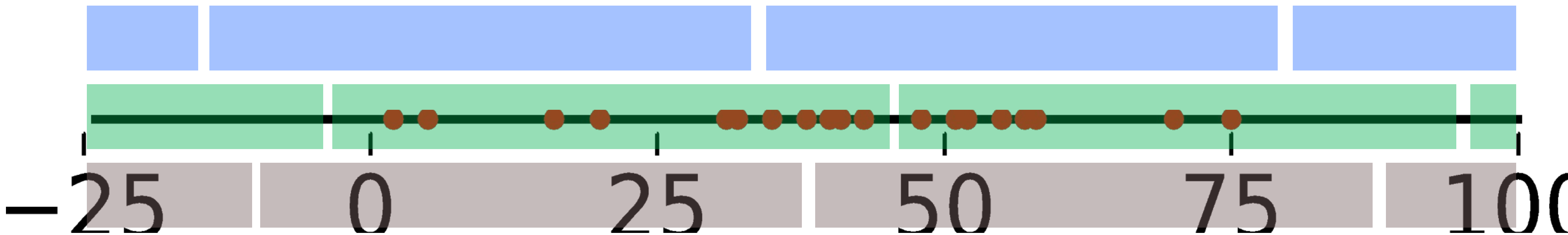
motif instance M

duration $d(M) = 32 - 16 = 16$

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.

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

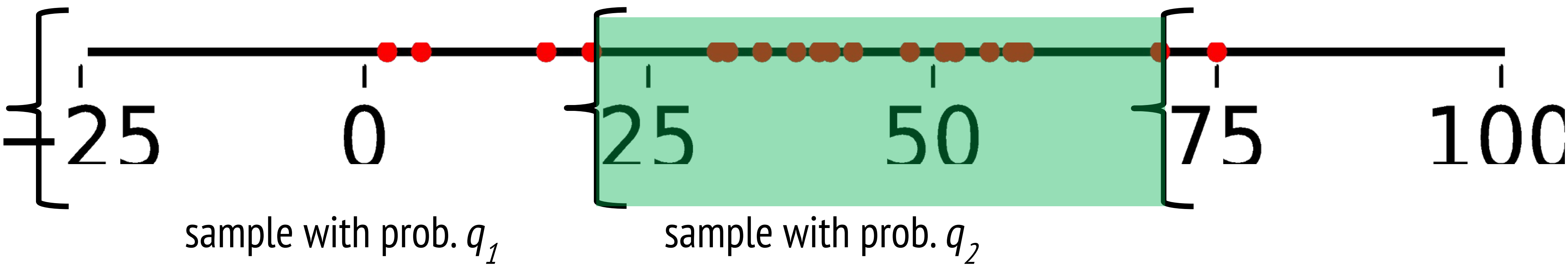
Sampling window length $w > \delta$ ($\delta = 25$, $w = 50$).
Choose random shift s uniformly from [0, 1, …, w - 1] (s = 20).



- 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.

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

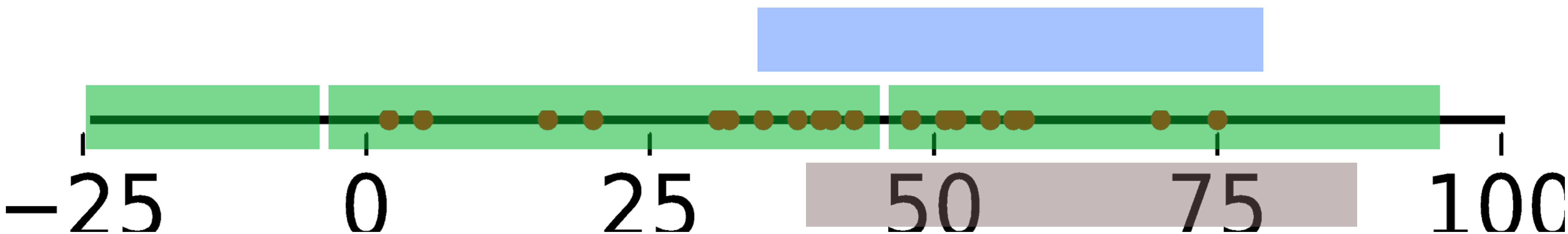
Sampling window length $w > \delta$ ($\delta = 25$, $w = 50$).
Choose random shift s uniformly from [0, 1, ..., w - 1] (s = 20).



- Set q_j for each shift. Larger $q_j \longrightarrow$ more computation but less variance.
- **Importance sampling.** Only want to sample where motifs occur.
- **Heuristic.** Make q_j larger if more edges in sampling window.

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.



Our algorithm in a nutshell.

Input. Temporal motif and maximum time scale δ .
Output. Estimate of number of instances of the motif.

Partition data into windows	Parallel Importance Sampling	Count and upscale
Choose a random shift *s*, and partition the data into windows aligned to *s*.	For *j*th window, sample with probability q_j.	Upscale counts of motif instances depending on their duration $d(M)$.

Our algorithm in a nutshell.

Input. Temporal motif and maximum time scale δ .
Output. Estimate of number of instances of the motif.

Partition data into windows → Parallel Importance Sampling → Count and upscale

Key advantages.

- Works in streaming setting. Faster & less memory intensive.
- Can use (almost) any "exact counting" method for step 3.
  [Paranjape-Benson-Leskovec 17; Mackey+ 18; Liu-Benson-Charikar 18]
- Can parallelize over shifts and sampling windows
  $\longrightarrow$ exposes parallelism to otherwise sequential algorithms.

THANKS!

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

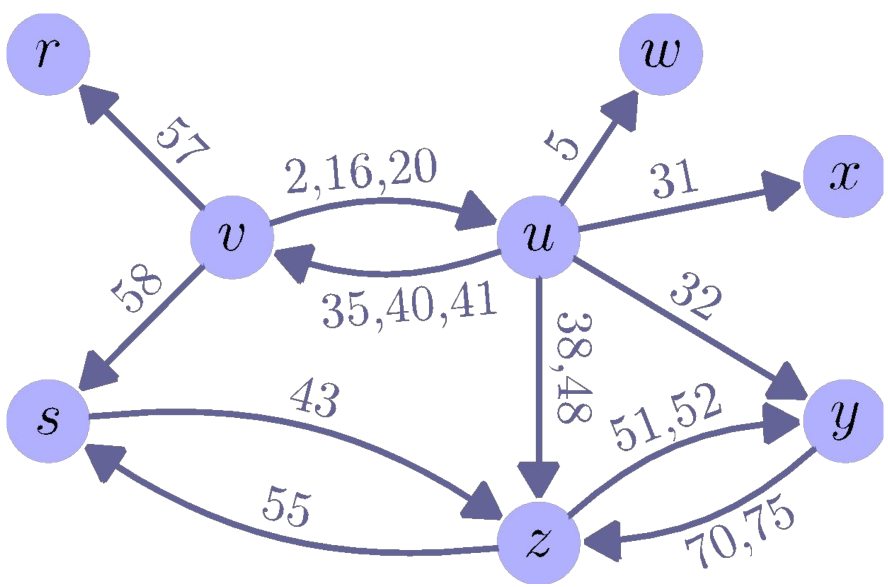
Cornell University

Stanford University

Existing methods for temporal analysis are insufficient.

1. **Models for network growth**
   Growth of academic collaborations, Internet infrastructure, etc. [Leskovec+ 07]

2. **Sequence of snapshot aggregates**
   Daily phone call graph [Araujo+ 14], weekly email snapshots [Xu-Hero 14]



Modern temporal network datasets

- fine-scale time resolution
- high-frequency
- many repeated edges