Efficient Streaming Algorithms for Graphlet Sampling

Yann Bourreau[1] Marco Bressan[2]
T-H. Hubert Chan[3] Qipeng Kuang[3] Mauro Sozio[4]

[1]CISPA, Saarland University

[2]University of Milan

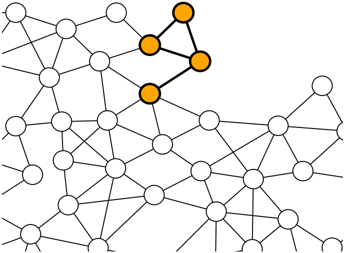
[3]The University of Hong Kong

[4]Institut Polytechnique de Paris, Télécom Paris

NEURAL INFORMATION PROCESSING SYSTEMS

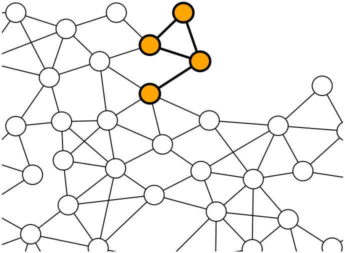
Graphlet Sampling (GS)

INPUT: a simple undirected graph G and $k \geq 3$
OUTPUT: a *uniform* random connected k-vertex subgraph of G (a k-*graphlet*)



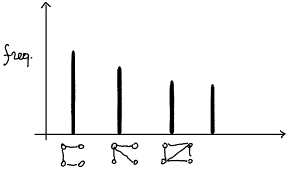
Graphlet Sampling (GS)

INPUT: a simple undirected graph G and $k \geq 3$
OUTPUT: a *uniform* random connected k-vertex subgraph of G (a k-*graphlet*)



Applications:

- sampling k-graphlets $\rightarrow$ k-graphlet distribution $\rightarrow$ feature vector



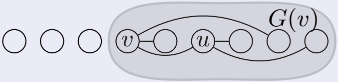
- graph classification, graph kernels, graph neural networks, clustered federated learning...

State-of-The-Art Algorithm

[Bressan, STOC 2021 and Algorithms 2023]

Two-phase uniform graphlet sampling

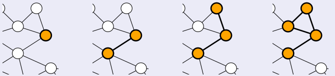
Preprocessing:



Degree-Dominating Order + Starting Probabilities

$O(n\,k^2 + m \log n)$ time

Sampling:



Random Grow + Rejection Sampling

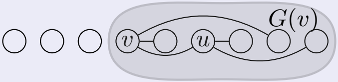
Expected $k^{O(k)} \log n$ time per graphlet

State-of-The-Art Algorithm

[Bressan, STOC 2021 and Algorithms 2023]

Two-phase uniform graphlet sampling

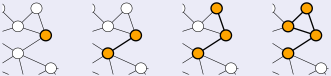
Preprocessing:



Degree-Dominating Order + Starting Probabilities

$O(n\,k^2 + m \log n)$ time

Sampling:



Random Grow + Rejection Sampling

Expected $k^{O(k)} \log n$ time per graphlet

Issue

Memory $O(m)$ – to store the whole graph

- E.g., Friendster: $n \approx 6.8 \times 10^7, m \approx 1.8 \times 10^9$.

Setting

Semi-Streaming Model

- Edges can only be accessed through *streaming passes*
  - 1 pass = scan the edge list sequentially in arbitrary order
- Memory $M = \tilde{O}(n)$

Goal: use a "small" number of passes

Setting

Semi-Streaming Model

- Edges can only be accessed through *streaming passes*
  - 1 pass = scan the edge list sequentially in arbitrary order
- Memory $M = \tilde{O}(n)$

Goal: use a “small” number of passes

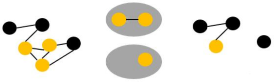
Challenges:

- Cannot store the graph in memory!
- Compute the order using $o(n)$ passes?

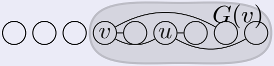
Our Results

Streaming Algorithm

Preprocessing:



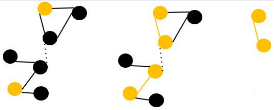
$\frac{1}{1+\epsilon}$-Degree-Dominating Order
$\tilde{O}(\log n)$ pass w.h.p.
$O(m \log n)$ time



Starting Probabilities

1 pass
$O(nk^2)$ time

Sampling:

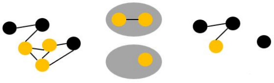


$\Theta(Mk^{-O(k)})$ parallel samples w.h.p.
$2k-1$ pass
$O(M2^k \log n + mk \log n)$ time

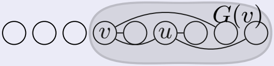
Our Results

Streaming Algorithm

Preprocessing:



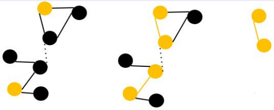
$\frac{1}{1+\epsilon}$-Degree-Dominating Order
$\tilde{O}(\log n)$ pass w.h.p.
$O(m \log n)$ time



Starting Probabilities

1 pass
$O(nk^2)$ time

Sampling:



$\Theta(Mk^{-O(k)})$ parallel samples w.h.p.
$2k - 1$ pass
$O(M2^k \log n + mk \log n)$ time

Streaming Lower Bound

For $k \geq 3$, any p-pass streaming algorithm for GS requires $\Omega\left({}^{n}/_{p}\right)$ bits of memory.

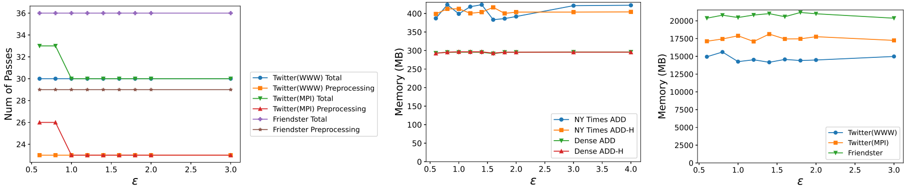
- Nearly space-optimal!

Experiments

Dataset [Kunegis, 2013]	File Size (MB)	#Vertices	#Edges
Dense	1,858	20,000	159,993,472
NY Times	858	401,388	69,654,798
Twitter (WWW)	20,437	41,652,230	1,202,513,047
Twitter (MPI)	25,590	52,579,682	1,614,106,188
Friendster	32,300	68,349,466	1,811,849,343

Experiments

Dataset [Kunegis, 2013]	File Size (MB)	#Vertices	#Edges
Dense	1,858	20,000	159,993,472
NY Times	858	401,388	69,654,798
Twitter (WWW)	20,437	41,652,230	1,202,513,047
Twitter (MPI)	25,590	52,579,682	1,614,106,188
Friendster	32,300	68,349,466	1,811,849,343



Number of passes and memory versus ϵ, fixing $k = 4$

Thank You!