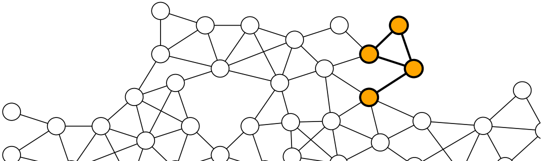
Linear and sublinear algorithms for graphlet sampling

Marco Bressan
Univ. of Milan

Hubert Chan
Hong Kong Univ.

Qipeng Kuang
Hong Kong Univ.

Mauro Sozio
Télécom Paris



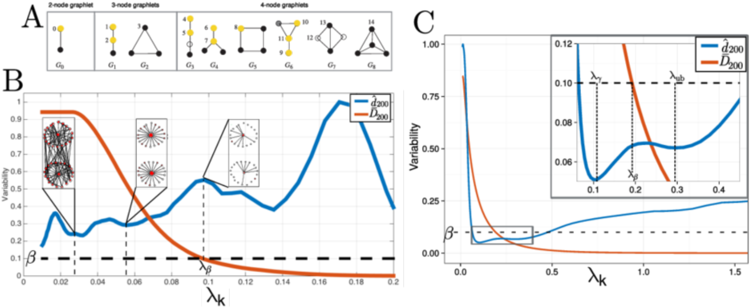
The Graphlet Sampling Problem

INPUT: a simple graph G and $k \geq 3$

OUTPUT: a uniform random connected k-vertex subgraph of G (a *k-graphlet*)

Applications:

- estimating the graphlet frequency vector
- network analysis, bioinformatics, clustering, ...



The Graphlet Sampling Problem

INPUT: a simple graph G and $k \geq 3$

OUTPUT: a uniform random connected k-vertex subgraph of G (a *k-graphlet*)

Many methods proposed ...

Bhuiyan et al. ICDM'12
Ahmed et al. TKDD'13
Ahmed et al. VLDB'14
Wang et al. TKDD'14
Saha et al. CompleNet'15
Jha et al. WWW'15
Bressan et al. WSDM'16
Han et al. ICDM'16
Chen et al. VLDB'16
Pinar et al. WWW'17
Bressan et al. TKDD'18
Agostini et al. IPL'19
Matsuno et al. SDM'20
Paramonov et al. KDD'20
...

... all with limitations:

- only for $k = 3, 4, 5$
- or, $n^{\Theta(k)}$ time per sample
- or, samples far from uniform
- ...

Results

Theorem 1 (the **linear** algo). There exists a two-phase uniform graphlet sampling algorithm with preprocessing time

$$\mathcal{O}(n\,k^2 \log k + m)$$

and expected sampling time per graphlet

$$k^{\mathcal{O}(k)} \log n$$

Theorem 2 (the **sublinear** algo). There exists a two-phase ε-uniform graphlet sampling algorithm with preprocessing time

$$\mathcal{O}\left(\varepsilon^{-1} k^6\, n \log n\right)$$

and expected sampling time per graphlet

$$k^{\mathcal{O}(k)} \varepsilon^{-10} \log \varepsilon^{-1}$$

Results

Theorem 1 (the **linear** algo). There exists a two-phase uniform graphlet sampling algorithm with preprocessing time

$$\mathcal{O}(n\,k^2 \log k + m) \quad \mathcal{O}(n + m)$$

and expected sampling time per graphlet

$$k^{\mathcal{O}(k)} \log n \quad \mathcal{O}(\log n)$$

Theorem 2 (the **sublinear** algo). There exists a two-phase ε-uniform graphlet sampling algorithm with preprocessing time

$$\mathcal{O}\left(\varepsilon^{-1} k^6\, n \log n\right) \quad \mathcal{O}(n \log n)$$

and expected sampling time per graphlet

$$k^{\mathcal{O}(k)} \varepsilon^{-10} \log \varepsilon^{-1} \quad \mathcal{O}(1)$$

Results

Theorem 1 (the **linear** algo). There exists a two-phase uniform graphlet sampling algorithm with preprocessing time

$$\mathcal{O}(n\,k^2 \log k + m) \quad \mathcal{O}(n+m)$$

and expected sampling time per graphlet

$$k^{\mathcal{O}(k)} \log n \quad \mathcal{O}(\log n)$$

Theorem 2 (the **sublinear** algo). There exists a two-phase ε-uniform graphlet sampling algorithm with preprocessing time

$$\mathcal{O}\left(\varepsilon^{-1} k^6\, n \log n\right) \quad \mathcal{O}(n \log n)$$

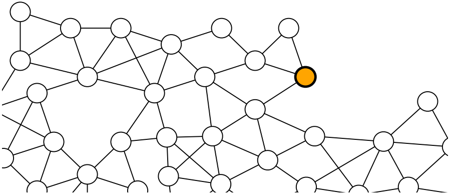
and expected sampling time per graphlet

$$k^{\mathcal{O}(k)} \varepsilon^{-10} \log \varepsilon^{-1} \quad \mathcal{O}(1)$$

Ongoing Work: streaming and MPC adaptations.

The Linear Algorithm

Starting Point: Rejection Sampling



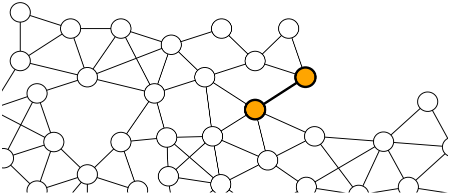
1. pick some vertex v in G with some probability $p(v)$
2. starting with $S = \{v\}$, while $|S| \leq k$ do *cut sampling*:

   pick an edge u.a.r. in $\delta(S)$ and add endpoint to S

   compute the probability $p(S|v)$ that cut sampling from v yields S

   let $p^* = \min_{v,S} p(v) \cdot p(S|v) > 0$
3. with probability $\frac{p^*}{p(v)\,p(S|v)}$ return S, else repeat from 1.

Starting Point: Rejection Sampling



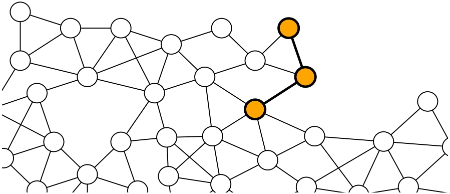
1. pick some vertex v in G with some probability $p(v)$
2. starting with $S = \{v\}$, while $|S| \leq k$ do *cut sampling*:

   pick an edge u.a.r. in $\delta(S)$ and add endpoint to S

   compute the probability $p(S|v)$ that cut sampling from v yields S

   let $p^* = \min_{v,S} p(v) \cdot p(S|v) > 0$
3. with probability $\frac{p^*}{p(v)\,p(S|v)}$ return S, else repeat from 1.

Starting Point: Rejection Sampling



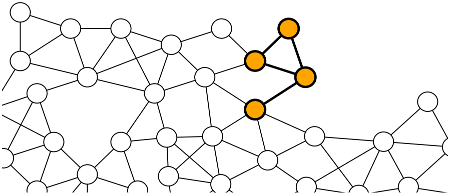
1. pick some vertex v in G with some probability $p(v)$
2. starting with $S = \{v\}$, while $|S| \leq k$ do *cut sampling*:

   pick an edge u.a.r. in $\delta(S)$ and add endpoint to S

   compute the probability $p(S|v)$ that cut sampling from v yields S

   let $p^* = \min_{v,S} p(v) \cdot p(S|v) > 0$
3. with probability $\frac{p^*}{p(v)\,p(S|v)}$ return S, else repeat from 1.

Starting Point: Rejection Sampling



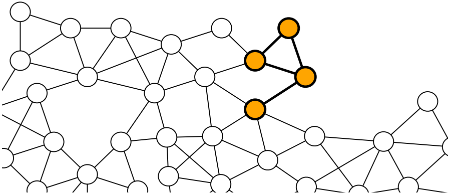
1. pick some vertex v in G with some probability $p(v)$
2. starting with $S = \{v\}$, while $|S| \leq k$ do *cut sampling*:

   pick an edge u.a.r. in $\delta(S)$ and add endpoint to S

   compute the probability $p(S|v)$ that cut sampling from v yields S

   let $p^* = \min_{v,S} p(v) \cdot p(S|v) > 0$
3. with probability $\frac{p^*}{p(v)\, p(S|v)}$ return S, else repeat from 1.

Starting Point: Rejection Sampling



1. pick some vertex v in G with some probability $p(v)$
2. starting with $S = \{v\}$, while $|S| \leq k$ do *cut sampling*:

   pick an edge u.a.r. in $\delta(S)$ and add endpoint to S

   compute the probability $p(S|v)$ that cut sampling from v yields S

   let $p^* = \min_{v,S} p(v) \cdot p(S|v) > 0$
3. with probability $\frac{p^*}{p(v)\, p(S|v)}$ return S, else repeat from 1.

$$\mathbb{P}[S \text{ returned}] = p(v) \cdot p(S|v) \cdot \frac{p^*}{p(v) \cdot p(S|v)} = p^*$$

samples are uniform :-)

$$\mathbb{P}[S \text{ returned} | S \text{ sampled}] = \frac{p^*}{p(v) \cdot p(S|v)}$$

can be VERY small, like $n^{-\Theta(k)}$:-(

How to Fix It

$$\mathbb{P}[S \text{ returned} | S \text{ sampled}] = \frac{p^*}{p(v) \cdot p(S|v)}$$ can be VERY small

We will make the sampling distribution **almost** uniform, i.e.:

$$\frac{\min_S p(v) \cdot p(S|v)}{\max_S p(v) \cdot p(S|v)} = k^{-O(k)}$$

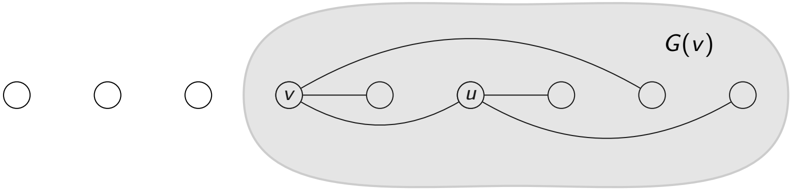
This will make

$$\mathbb{P}[S \text{ returned} | S \text{ sampled}] \geq k^{-O(k)} \qquad \forall S$$

I promise that everything will be efficient.

How to Fix It: Preprocessing

Compute a total order $\prec$ over V by repeatedly removing a vertex of max degree. This takes time $\mathcal{O}(n+m)$.



For every v define the subgraph

$$G(v) := G[u \succeq v]$$

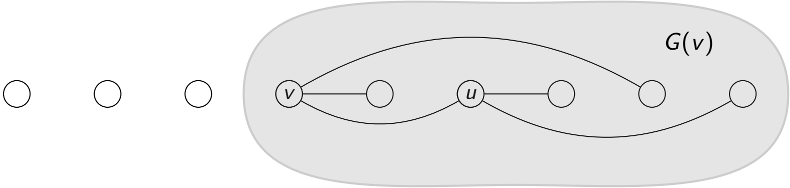
and the related **bucket** of graphlets

$$B(v) = \{S \text{ graphlet in } G(v),\ v \in S\}$$

Finally, for all v and all $u \succeq v$ define

$$d(u|v) = \text{ degree of } u \text{ in } G(v)$$

How to Fix It: Sampling



Obs 1. Every nonempty bucket $B(v)$ satisfies:

$$|B(v)| \overset{k^{O(k)}}{\simeq} d(v|v)^{k-1}$$

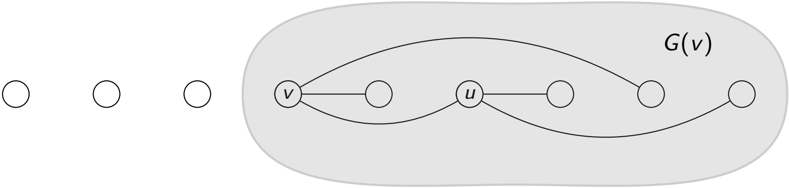
(Proof by counting argument.)

Then, we set

$$p(v) = \frac{d(v|v)^{k-1}}{Z} \overset{k^{O(k)}}{\simeq} \frac{|B(v)|}{Z}$$

where $Z = \sum_u d(u|u)^{k-1}$.

How to Fix It: Sampling



Obs 2. Suppose we run the cut sampling process over $G(v)$ starting from v. Then any $S \in B(v)$ is sampled with probability:

$$p(S|v) \overset{k^{O(k)}}{\simeq} \frac{1}{|B(v)|} \quad \Rightarrow \quad p(v) \cdot p(S|v) \overset{k^{O(k)}}{\simeq} \frac{1}{Z}$$

(Proof by counting argument.)

Obs 3. If we sort the adjacency lists of G by $\prec$, then the cut sampling process over $G(v)$ can be run in time $\mathrm{poly}(k) \log n$.

Proof: for every $u \in G(v)$ we can locate the cut $E(u, G(v))$ by binary searching for v in u's adjacency list.

The Uniform Algorithm

Algorithm 1 UGS

procedure PREPROCESSING
    compute the total order $\prec$ over V
    compute $p(v) := \frac{d(v|v)^{k-1}}{Z}$ for all v, where $Z = \sum_u d(u|u)^{k-1}$

procedure SAMPLING
    sample v with probability $p(v) := \frac{d(v|v)^{k-1}}{Z}$
    let $S = \{v\}$
    grow S in $G(v)$ via cut sampling until $|S| = k$
    compute $p(S|v)$
    return S with probability $\frac{k^{-O(k)}/Z}{p(v) \cdot p(S|v)}$

The Uniform Algorithm

Algorithm 2 UGS

procedure PREPROCESSING
    compute the total order $\prec$ over V
    compute $p(v) := \frac{d(v|v)^{k-1}}{Z}$ for all v, where $Z = \sum_u d(u|u)^{k-1}$
    set $p(v)$ to 0 if $\mathbb{I}\{B(v) = \emptyset\}$, for all v

procedure SAMPLING
    sample v with probability $p(v) := \frac{d(v|v)^{k-1}}{Z}$
    let $S = \{v\}$
    grow S in $G(v)$ via cut sampling until $|S| = k$
    compute $p(S|v)$
    return S with probability $\frac{k^{-O(k)}/Z}{p(v) \cdot p(S|v)}$

The Uniform Algorithm

Algorithm 3 UGS

procedure PREPROCESSING
  compute the total order $\prec$ over V — $\mathcal{O}(n+m)$
  compute $p(v) := \frac{d(v|v)^{k-1}}{Z}$ for all v, where $Z = \sum_u d(u|u)^{k-1}$ — $\mathcal{O}(n)$
  set $p(v)$ to 0 if $\mathbb{I}\{B(v) = \emptyset\}$, for all v — $\mathcal{O}(nk^2 \log k)$

procedure SAMPLING
  sample v with probability $p(v) := \frac{d(v|v)^{k-1}}{Z}$ — $\mathcal{O}(1)$
  let $S = \{v\}$
  grow S in $G(v)$ via cut sampling until $|S| = k$ — $\mathcal{O}(k^3 \log n)$
  compute $p(S|v)$ — $k^{\mathcal{O}(k)} \log n$
  return S with probability $\frac{k^{-O(k)}/Z}{p(v) \cdot p(S|v)}$ — $\mathcal{O}(1)$

The Uniform Algorithm

Algorithm 4 UGS

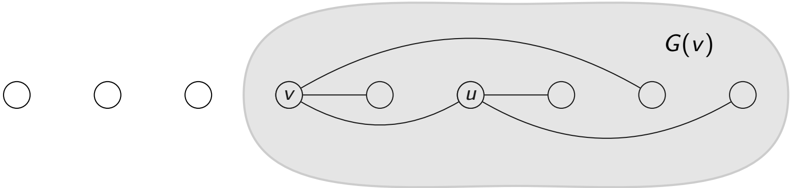
procedure PREPROCESSING
    compute the total order $\prec$ over V — $\mathcal{O}(n+m)$
    compute $p(v) := \frac{d(v|v)^{k-1}}{Z}$ for all v, where $Z = \sum_u d(u|u)^{k-1}$ — $\mathcal{O}(n)$
    set $p(v)$ to 0 if $\mathbb{I}\{B(v) = \emptyset\}$, for all v — $\mathcal{O}(nk^2 \log k)$

procedure SAMPLING
    sample v with probability $p(v) := \frac{d(v|v)^{k-1}}{Z}$ — $\mathcal{O}(1)$
    let $S = \{v\}$
    grow S in $G(v)$ via cut sampling until $|S| = k$ — $\mathcal{O}(k^3 \log n)$
    compute $p(S|v)$ — $k^{\mathcal{O}(k)} \log n$
    return S with probability $\frac{k^{-O(k)}/Z}{p(v) \cdot p(S|v)}$ — $\mathcal{O}(1)$

Total preprocessing time: $\mathcal{O}(nk^2 \log k + m)$

Expected sampling time: $k^{\mathcal{O}(k)} \log n$

The Sublinear Algorithm



Definition. A total order $\prec$ over V is α-degree-dominating (α-DD) if

$$d(v|v) \geq \alpha \cdot d(u|v) \qquad \forall v \quad \forall u \in G(v)$$

That is, v has *approximately* the largest degree in $G(v)$.

Naive attempt for the ε-uniform algo:

1. compute an ε-DD ordering $\prec$ in time $\mathcal{O}(\varepsilon^{-1} n \log n)$
2. sample as before

The ε-Uniform Algorithm: Preprocessing

We need the following relaxation.

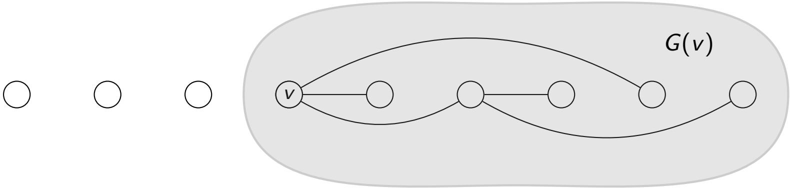
An (α, β)-DD order for G is a pair $(\prec, \boldsymbol{b})$ where $\boldsymbol{b} = (b_v)_{v \in V}$ such that:

(1) $b_v > 0 \implies d(v|G(v)) \geq \alpha\, d_v \geq \alpha\, d(u|G(v))$ for all $u \succ v$

(2) $b_v > 0 \implies k^{-\mathcal{O}(k)}\beta \leq \frac{b_v}{|B(v)|} \leq k^{\mathcal{O}(k)}\frac{1}{\beta}$

(3) $\sum_{v:b_v=0} |B(v)| \leq \beta \sum_v |B(v)|$

(4) $v \prec u \implies d_u \leq \frac{d_v}{3k\alpha}$



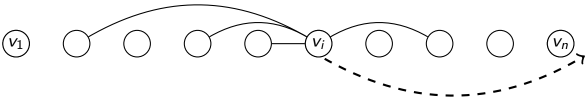
The ε-Uniform Algorithm: Preprocessing

Theorem. In time $\mathcal{O}\left(\beta^{-1} k^6\, n \log n\right)$ one can compute with high probability an (α, β)-DD order $(\prec, \boldsymbol{b})$ with $\alpha = \beta^{\frac{1}{k-1}} \frac{1}{6k^3}$.

Proof is really unwieldly.

Algorithm 5 Compute-(α, β)-DD

```
start with v_1, ..., v_n in nonincreasing order of degree
for i = 1, ..., n do
    draw O(β^{-2} k^4 log n) random neighbors of v_i
    if a fraction ≥ β/k^2 of those neighbors are after v_i then
        set b_v = d_v^{k-1}
    else
        set b_v = 0 and push v_i at the end of the order
```



The ε-Uniform Algorithm: Sampling

We compute an (α, ε)-DD order $(\prec, \boldsymbol{b})$ in time $\mathcal{O}\big(\varepsilon^{-1} k^6\, n \log n\big)$.

Then, we run the sampling phase of UGS using $(\prec, \boldsymbol{b})$.

As the sampling phase guarantees uniformity over the support of the distribution, by (3) we'll get ε-uniform graphlets.

REMINDER – Properties of (α, β)-DD ordering:
(3) $\sum_{v:b_v=0} |B(v)| \leq \beta \sum_v |B(v)|$

Problem: without sorted adjacency lists, cut sampling can take time $\Omega(n)$.

The ε-Uniform Algorithm: Sampling

How to fix it:

1. Set $\beta = \frac{\varepsilon}{2}$. By (3) we only need $\frac{\varepsilon}{2}$-uniformity over the $B(v)$ s.t. $b_v > 0$.
2. When growing S in $G(v)$, estimate the cut $E(u, G(v) \setminus S)$ of every $u \in S$. By (1),(4) we can estimate $E(u, G(v) \setminus S)$ with good multiplicative accuracy. This ensures $p(S|v)$ somewhat close to $\frac{1}{|B(v)|}$.
3. Use cut estimates to approximately compute $p(S|v)$.
4. Finally, by (2) the acceptance probability is not much worse than in $\textsc{Ugs}$.

REMINDER – Properties of (α, β)-DD ordering:

(1) $b_v > 0 \implies d(v|G(v)) \geq \alpha\, d_v \geq \alpha\, d(u|G(v))$ for all $u \succ v$

(2) $b_v > 0 \implies k^{-\mathcal{O}(k)}\beta \leq \frac{b_v}{|B(v)|} \leq k^{\mathcal{O}(k)}\frac{1}{\beta}$

(3) $\sum_{v: b_v = 0} |B(v)| \leq \beta \sum_v |B(v)|$

(4) $v \prec u \implies d_u \leq \frac{d_v}{3k\alpha}$

The ε-Uniform Algorithm

Algorithm 6 APX-UGS

procedure PREPROCESSING
    compute an $(\alpha, \frac{\varepsilon}{2})$-order $(\prec, \boldsymbol{b})$ over V — $\mathcal{O}(\varepsilon^{-1} k^6 n \log n)$
    compute $p(v) := \frac{b_v}{Z}$ for all v, where $Z = \sum_u b_v$ — $\mathcal{O}(n)$

procedure SAMPLING
    sample v with probability $p(v) := \frac{b_v}{Z}$ — $\mathcal{O}(1)$
    let $S = \{v\}$
    grow S in $G(v)$ via cut sampling until $|S| = k$ — $k^{\mathcal{O}(k)} \varepsilon^{-8} \log \varepsilon^{-1}$
    compute an estimate $\widehat{p}(S|v)$ of $p(S|v)$ — $k^{\mathcal{O}(k)} \varepsilon^{-8} \log \varepsilon^{-1}$
    return S with probability $\frac{\varepsilon\, k^{-O(k)}/Z}{p(v) \cdot \widehat{p}(S|v)}$ — $\mathcal{O}(1)$

Total preprocessing time: $\mathcal{O}(\varepsilon^{-1} k^6 n \log n)$

Expected sampling time: $k^{\mathcal{O}(k)} \operatorname{poly}(\varepsilon^{-1})$

Extensions: Streaming and MPC

Streaming and MPC

Streaming: adversarial stream of edges, memory of $M = \Omega(n \log n)$ words.

Theorem. The following guarantees are achievable w.h.p.:

	# passes	running time	# uniform samples
preprocessing	$O(\log n)$	$O(p\,m + n\,k^2 \lg k)$	–
*sampling**	$2k$	$O((n\,2^k + m\,k) \log n)$	$\Omega(M \cdot k^{-O(k)})$

MPC: M machines, $S \geq M$ words per machine, $M \cdot S = \tilde{\Omega}(n + m)$.

Theorem. The following guarantees are achievable w.h.p.:

	# rounds	running time	# uniform samples
preprocessing	$k + O(\log n \log \log n)$	$k^{O(k)}(n + m)$	–
*sampling**	$O(k)$	$k^{O(k)}(n + m)$	$\Omega(M \cdot S \cdot k^{-O(k)})$

Open Problems

(1) What can we achieve without preprocessing?

(2) What about **hypegraphs**?