Estimating Graphlet Statistics via Lifting

Kirill Paramonov
kir.paramonov@gmail.com
University of California, Davis

James Sharpnack
jsharpna@gmail.com
University of California, Davis

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

ACM Reference Format:
Kirill Paramonov and James Sharpnack. 2018. Estimating Graphlet Statistics via Lifting. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, **??** pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In 1970, [**?**] discovered that transitivity—friends of friends tend to be friends themselves—is a prevalent feature in social networks. Since that early discovery, real-world networks have been observed to have many other common macroscopic features, and these discoveries have lead to probabilistic models for networks that display these phenomena. The observation that transitivity and other common subgraphs are prevalent in networks lead to the exponential random graph model [**?**]. [**?**] demonstrated that many large networks display a scale-free power law degree distribution, and provided a model for constructing such graphs. Similarly, the small world phenomenon—that networks display surprisingly few degrees of separation—motivated the network model in [**?**]. While network science is often driven by the observation and modelling of common properties in networks, it is incumbent on the practicing data scientist to explore network data using statistical methods.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

One approach to understanding network data is to fit free parameters in these network models to the data through likelihood-based or Bayesian methods. In [**?**], a pseudolikelihood method was used with graphlet counts to fit an ERGM designed to display transitivity, and Monte Carlo Markov Chain methods were developed in [**?**] for fitting general ERGMs. Fitting such models from data can be cumbersome, and to do so implicitly assumes that the network follows such a model exactly. Network statistics, such as the clustering coefficient, algebraic connectivity, and degree sequence, are more flexible tools. A good statistic can be used to fit and test models, for example, [**?**] used the local clustering coefficient, a measure of the number of triangles relative to wedges, to test if a network is a small-world graph. The clustering coefficient is also used to understand social network graphs, [**?**]. More generally, it was discovered that re-occurring subgraph patterns can be used to differentiate real-world networks, and that genetic networks, neural networks, and internet networks all presented different common interconnectivity patterns, [**?**]. In this work, we will propose a new method for counting the occurrences of any subgraph pattern, otherwise known as *graphlets*—a term coined in [**?**]—or motifs.

A graphlet is a small graph topology, such as a triangle, wedge, or k-clique, which we will use to describe the local behavior of a larger network (example graphlets of size 3, 4, and 5, can be seen in Figure **??**). Let the graph in question be $G = (V, E)$ where V is a set of vertices and E is a set of unordered pairs of vertices (G is assumed to be undirected). Imagine specifying a k-graphlet and testing for every induced subgraph of the graph (denoted $G|\{v_1, \ldots, v_k\}$ where $v_1, \ldots, v_k \in V$), if it is isomorphic (has the same topology) to the subgraph. We would like to compute the number of subgraphs for which this match holds, and the proportion of the number of such matches to the number of connected induced subgraphs of that size is called the *graphlet coefficient*.

Graphlets are the graph analogue of wavelets (small oscillatory functions that are convolved with a signal to produce wavelet coefficients) because they are small topologies that are matched to induced subgraphs of the original graph to produce the graphlet coefficients. Graphlet coefficients, also referred to as graph moments, are used in the method of moments to fit certain graph models by selecting parameters that match the empirical graph moments to their expectations [**?**]. Graphlet coefficients are used to understand biological networks, such as the protein-protein interaction network, and reoccuring patterns are thought to indicate evolutionary conserved modules [**?**]. If k is small then testing for isomorphism, a problem called graph matching [**?**], is feasible, but testing every induced subgraph can require on the order of n^k iterations in its most naive

implementation. We propose a class of methods called lifting methods that allow us to quickly estimate graphlet coefficients.

While there exist several algorithms that can estimate the proportion of triangles, wedges, and graphlets with four or five vertices (for eaxample, [**?** **?**]), there are few algorithms that can efficiently estimate the proportion of larger graphlets. Many methods that can handle arbitrary graphlets are Monte Carlo sampling procedures that traverse through the space of all subgraphs of a certain size. Two such methods are GUISE algorithm of [**?**], the pairwise subgraph random walk of [**?**], and the Waddling random walk of [**?**], which differ in the way that they perform a random walk between the induced subgraphs. Alternatives to Monte Carlo schemes include the color coding scheme of [**?**], but it processes the whole graph, while the Monte Carlo schemes can traverse the network locally. In this work, we propose a new Monte Carlo algorithm, called lifting, for estimating the graphlet frequencies within a large network. The lifting step takes a smaller subgraph of size $k-1$ and produces a subgraph of size k by adding an adjacent vertex to it (according to a specific scheme). In this paper we consider procedures that start from vertices or edges and lift to sample subgraphs of size k. Lifting is a simple, flexible procedure that can be easily distributed to accommodate massive network datasets.

1.1 Our contributions

Graphlet coefficients are multipurpose statistical tools that can be used for model fitting and testing, network regression, and exploratory analysis for network data. Any subgraph Monte Carlo sampling scheme has three goals: that it provides unbiased estimates of the graphlet coefficients, that the variance of the estimated coefficients is small, and that it does so in as few iterations as possible. Because massive graphs are often stored in distributed databases, we would like the sampling scheme to require only neighborhood queries (requests to the database returns the neighbors of a node) and we will avoid storing or processing the full graph. Because communication is often the bottleneck in distributed computing, neighborhood queries are the basic unit for measuring computational time complexity.

After discussing the precise requirements for any graphlet sampling procedure, we will introduce the lifting scheme for graphlets. The key difficulty in any Monte Carlo method for graphlet coefficient is calculating the sampling distribution. We provide two methods, the ordered lift estimator and the unordered lift estimator, which differ in the way that subgraphs are represented and counted in the graphlet coefficient. The ordered estimator allows for a modification, called *shotgun sampling* that samples multiple graphlets in one shot. For our theoretical component, we prove that the estimated graphlet coefficients are unbiased when the underlying MCMC has reached the stationary distribution (called perfect mixing). We also prove that under perfect mixing, the variance of the estimator scales like Δ^{k-2} where Δ is the maximum degree, and show that the lifting scheme can have significantly lower mixing rates than the subgraph random walk. We conclude with real-world network experiments that reinforce the contention that graphlet lifting has a lower variance and better mixing rates than subgraph random walks.

2 ESTIMATING GRAPHLET FREQUENCIES VIA SAMPLING

2.1 Definitions and notation

Consider graph $G = (V, E)$ with set of vertices V and set of edges $E \subseteq \binom{V}{2}$. We assume G to be simple, connected and undirected. (It's easy to translate our results to the case of directed graphs, since we will mostly be focusing on sampling procedure.) For a subset $W \subseteq V$, a subgraph of G induced by W is a graph with vertices in W and edges in $\binom{W}{2} \cap E$. Notation: $G|W$.

If $|W| = k$ and $G|W$ is connected, then we say that $G|W$ is a k-graphlet of G. The set of all k-graphlets of G is denoted by $\mathcal{V}_k(G)$ (or simply $\mathcal{V}_k$). Given a connected graph H on k nodes, we'll be interested in the number of k-graphlets of G isomorphic to H.

We'll make use of Python notation for lists vs sets: if we don't need indices or respective ordering of elements, we'll write it as a set $\{v_1, \ldots, v_k\}$, and if the order of the elements is fixed or indices are used to distinguish the elements, we'll write it as a list $[v_1, \ldots, v_k]$.

Let $Gr_k = [H_1, H_2, \ldots, H_I]$ be the list of all non-isomorphic connected graphs on k vertices. For $T \in \mathcal{V}_k(G)$, we say that "T is graphlet of type m" if T is isomorphic to H_m. Notation: $T \sim H_m$.

The number of k-graphlets of G of type m is equal to $N_m(G) = \sum_{T \in \mathcal{V}_k(G)} \mathbb{1}(T \sim H_m)$, where $\mathbb{1}(A)$ is the indicator function of boolean A.

2.2 Monte Carlo estimation of graphlet frequencies

Monte Carlo Markov Chains (MCMC) typically start from an initial distribution that then converges to the stationary distribution. A requirement

The ideal Monte Carlo procedure would sequentially sample graphlets uniformly at random from the set of all graphlets within G, classify their type, and update the corresponding moments. Unfortunately, uniformly sampling graphlets is not a simple task because they are required to be connected—a random set of k vertices is unlikely to be connected. We turn to Markov chains to sample more effectively. First, let us consider how we update the graph moments, $N_m(G)$, given a sample of graphlets $T_1, T_2, \ldots, T_n$.

The desire to sample graphlets uniformly is natural, because the empirical moments will be unbiased estimates of the graph moments. Suppose that each graphlet, T_i, is drawn from the common distribution π over the space of graphlets. Then we use Horvitz-Thompson inverse probability weighting to estimate

the graph moments,

$$\hat{N}_m(G) := \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbb{1}(T_i \sim H_m)}{\pi(T_i)}. \quad (1)$$

It is simple to see that this is an unbiased estimate of the moments as long as π is supported over all k-graphlets. We find ourselves in a game, where we can choose any graphlet Monte Carlo algorithm that induces a stationary distribution π, but we must be able to quickly and accurately compute π in order to use (**??**). We will analyze the theoretical implications of the sampling algorithm based on mixing times of the Markov chain and the variance of the moment estimates in Section ??.

2.3 Subgraph random walk

Before we explore the lifting procedure, this paper's algorithmic contribution, we would like to discuss some existing Monte Carlo graphlet sampling methods. Any admissible graphlet Markov chain transitions between k-graphlets of G and the transition probabilities have a calculable stationary distribution π. The basic algorithm is listed in **??**, and some elements are left purposefully vague to maintain generality. Many of the proposed graphlet sampling algorithms can be encompassed by this master algorithm, and we will highlight the two most prominent examples: Subgraph random walk and waddling random walk.

Algorithm 1 Subgraph Markov chain

Initialize T at an arbitrary k-graphlet, $n \leftarrow 1$, $\hat{N}_m(G) \leftarrow 0$
while stopping criteria is not met **do**
  Query the neighbors of each vertex in T, $\{\mathcal{N}(v) : v \in T\}$
  Add a vertex according to an *add rule*
  Remove a vertex according to a *remove rule*
  if *Markov chain is sufficiently mixed* **then**
    Determine the type m of T, calculate $\pi(T)$
    Update $\hat{N}_m(G) \leftarrow ((n-1)\hat{N}_m(G) + \pi^{-1}(T))/n$ and $n \leftarrow n+1$
  end if
end while

Mixing is a critical issue for any Markov chain based sampling, and graphlet sampling is no exception. We will mathematically define mixing in Section ??, but, in words, it measures the dependence between consecutive samples. Dependence between consecutive samples results in a higher variability of $\hat{N}_m(G)$, and because calculating $\pi(T)$ and determining the type of T can be computationally cumbersome, it is advisable to not include every subgraph in the final sample. One easy way to achieve this is to sample only every several iterations, where the sampling frequency is calibrated to obtain a certain level of mixing.

Subgraph random walk.

2.4 Lifting procedures

This approach gives an unbiased estimator assuming $\pi_k(T) > 0$ for any $T \in \mathcal{V}_k$. The biggest issue of this method is the variance of $\hat{N}_m(G)$. We rewrite the variance as a sum of two terms:

$$\mathrm{Var}(\hat{N}_m(G)) = \frac{1}{n}\left(\mathbb{E}_{\pi_k}\frac{\mathbb{1}(T \sim H_m)}{\pi_k(T)^2} - N_m^2(G)\right) + \frac{2}{n^2}\sum_{i<j}\mathrm{Cov}_{\pi_k}\left(\frac{\mathbb{1}(T_i \sim H_m)}{\pi_k(T_i)}, \frac{\mathbb{1}(T_j \sim H_m)}{\pi_k(T_j)}\right). \quad (2)$$

The first term will be referred to as variation under independence term, and can dominate the sum if $\pi_k(T)$ greatly deviates from the uniform probability $\rho(T) = \frac{\mathbb{1}(T \sim H_m)}{N_m(G)}$. The second term will be referred to as a covariance term and can dominate the sum when graphlets T_i and T_{i+1} have high correlation, which is the case for Subgraph Random Walk methods discussed below.

The previous papers on that topic focused on efficient method of sampling a sequence of k-graphlets, depending on a framework. We'll try to summarize the frameworks and corresponding methods here.

There are two different frameworks, each with its own problem statement.

- **Framework 1.** We have access to the whole graph structure, for example, in the form of adjacency matrix. We can sample the nodes of the graph cheaply, independently and according to an arbitrary probability distribution. The goal is to sample graphlets quickly, uniformly and independently. In particular, the covariance term would be irrelevant in this framework. The state-of-art method for this framework is Color Coding [**?**].
- **Framework 2.** We start with an access to a single node v_0 and its neighborhood. As a query, we can transition to a neighboring node v_1 and get access to the neighborhood of v_1. Each moment of time we can only get access to the neighbors of previously visited nodes. This framework is applied when queries are either expensive or limited and traversing the whole graph is time-prohibitive at best and not feasible at worst. The goal here is to achieve best accuracy with either limited number of queries (10^3-10^4 queries for graphs with 10^5-10^6 nodes) or limited time, or something in between (costly queries). Recent methods include Subgraph Random Walk(SRW) and Pairwise Random Walk [**?**] and Waddling Random Walk (WRW) [**?**].

For each of the two frameworks above we'll present a modification of the base algorithm to suit corresponding goals.

2.5 Subgraph Random Walk (SRW)

For this method, consider a set $\mathcal{G}_k$ of all k-graphlets of G. We introduce graph structure on $\mathcal{G}_k$ by connecting two graphlets $T, S \in \mathcal{G}_k$ with an edge if and only if vertex sets of T and S differ by one element, i.e. when $|V(T) \cap V(S)| = k-1$. Given the graph structure, we sample k-graphlets by random walk on the set of vertices $\mathcal{G}_k$, which is called Subgraph Random Walk (SRW). It was pointed out in [**?**] that the mixing time of the SRW can be of order $O(n^{k-2})$, even if the mixing time of the random walk on the original graph G is of constant

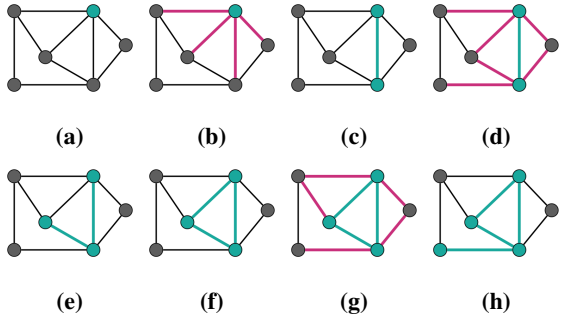

Figure 1: Lifting procedure

order $O(1)$. Thus, it would require $O(n^{k-2})$ burn-in steps to get uncorrelated samples from SRW.

3 ALGORITHM DESCRIPTION (TO BE MERGED WITH ABOVE)

3.1 Lifting procedure

For a subgraph $S \subseteq G$, denote V_S to be the set of its vertices, E_S to be the set of its edges. Denote $\mathcal{N}_v(S)$ (vertex neighborhood of S) to be the set of all vertices adjacent to some vertex in S not including S itself. Denote $\mathcal{N}_e(S)$ (edge neighborhood of S) to be the set of all edges that connect a vertex from S and a vertex outside of S. Formally,

$$\mathcal{N}_v(S) = \{u \in V_G \setminus V_S \mid \exists s \in V_S : (s,u) \in E_G\},$$

$$\mathcal{N}_e(S) = \{(s,u) \in E_G \mid s \in V_S, u \notin V_S\}.$$

Denote $\deg(S)$ (degree of S) to be the number of edges in $\mathcal{N}_e(S)$, and denote $\deg_S(u)$ (S-degree of u) to be the number of vertices from S that are connected to u. Formally,

$$\deg(S) = |\{(s,u) \in E_G \mid s \in V_S, u \notin V_S\}|,$$

$$\deg_S(u) = |\{s \in V_S \mid (u,s) \in E_G\}|.$$

Note that $\deg(S) + 2|E_S| = \sum_{v \in V_S} \deg(v)$.

First assume we have a node v_1 sampled from the distribution π_1 (we will address this assumption later). We also assume that $\pi_1(v) = \frac{f(\deg(v))}{K}$, where $f(x)$ is some function (usually a polynomial) and K is some global normalizing constant. Denote $S_1 = \{v_1\}$.

To start our procedure, sample an edge (v_1, v_2) uniformly from $\mathcal{N}_e(S_1)$. The vertex v_2 is then attached to S_1, forming a subgraph $S_2 = G|(V_{S_1} + v_2)$. After that, we sample another edge (v_i, v_3) (with $1 \le i \le 2$) uniformly from $\mathcal{N}_e(S_2)$, and the vertex v_3 is then attached to S_2.

Generally, sample an edge (v_i, v_{r+1}) (with $1 \le i \le r$) from $\mathcal{N}_e(S_r)$ uniformly at random, and attach the vertex v_{r+1} to the subgraph S_r. After $k-1$ operations, we obtain a k-graphlet $T = S_k$.

We'll refer to the procedure above as the **lifting** procedure starting at vertex v_1.

Using the lifting procedure, we can get two estimators of the graphlet count statistic.

3.2 Ordered lift estimator

We can think of a lifting procedure as a way of sampling a sequence $A = [v_1, \dots, v_k]$ that is then used to generate a graphlet. Denote the set of such sequences as V_G^k. For $A = [v_1, \dots, v_k] \in V_G^k$, denote $A[i] := v_i$. Let $S_r = S_r(A) = G|\{v_1, \dots, v_r\}$ be the r-graphlet obtained by the lifting procedure on step r. The probability of sampling vertex v_{r+1} on the step $r+1$ is equal to

$$\mathbb{P}(v_{r+1}|S_r) := \frac{\deg_{S_r}(v_{r+1})}{|\mathcal{N}_e(S_r)|} = \frac{|E_{S_{r+1}}| - |E_{S_r}|}{\sum_{i=1}^{r} \deg(v_i) - 2|E_{S_r}|}.$$

THIS IS KEY STEP. Thus, the probability of sampling a sequence $A \in V_G^k$ is equal to

$$\tilde{\pi}_k(A) := \pi_1(A[1]) \prod_{r=1}^{k-1} \mathbb{P}\left(A[r+1]|S_r(A)\right) =$$

$$\frac{f(\deg(A[1]))}{K} \prod_{r=1}^{k-1} \frac{|E_{S_{r+1}(A)}| - |E_{S_r(A)}|}{\sum_{i=1}^{r} \deg(A[i]) - 2|E_{S_r(A)}|}. \quad (3)$$

Consider the sampled k-graphlet $T := S_k(A)$. Denote the set of possible sequences $A = [v_1, \dots, v_k]$ that would form a graphlet T in the lifting process as $\mathrm{co}(T)$. Notice that $S_r(A) = G|\{v_1, \dots, v_r\} = T|\{v_1, \dots, v_r\}$ must be a connected subgraph for all r. Thus,

$$\mathrm{co}(T) = \Big\{ [v_1, \dots, v_k] \in V_G^k \mid \{v_1, \dots, v_k\} = V_T, \\ T|\{v_1, \dots, v_r\} \text{ is connected} \Big\}.$$

Since the elements of $\mathrm{co}(T)$ are just certain orderings of vertices in T, we call an element from $\mathrm{co}(T)$ a *compatible ordering* of T. Note that $|\mathrm{co}(T)|$ only depends on the type of the graphlet T. Thus, when $T \sim H_m$, the number of compatible orderings are equal: $|\mathrm{co}(H_m)| = |\mathrm{co}(T)|$. Note that $|\mathrm{co}(H_m)|$ can vary from 2^{k-1} (for k-path) to $k!$ (for k-clique).

Next, we set up the expectation:

$$\frac{1}{|\mathrm{co}(H_m)|} \mathbb{E}_{\tilde{\pi}_k} \frac{\mathbb{1}\left(T \sim H_m\right)}{\tilde{\pi}_k([v_1, \dots, v_k])} = \sum_{[v_1, \dots, v_k]} \frac{\mathbb{1}(T \sim H_m)}{|\mathrm{co}(T)|} = \\ \sum_{T} \mathbb{1}(T \sim H_m) = N_m(G).$$

The empirical expectation of the value on the left-hand side gives an estimate for $N_m(G)$. Let $A_1, \dots, A_n$ be sequences from V_G^k sampled in the lifting process. Then

$$\frac{1}{n} \frac{1}{|\mathrm{co}(H_m)|} \sum_i \frac{\mathbb{1}(G|A_i \sim H_m)}{\tilde{\pi}_k(A_i)} \approx N_m(G).$$

We call that estimator an *ordered lift estimator* for graphlet count statistic.

A drawback of the algorithm is that it takes $k-1$ queries to lift the graphlet plus the number of steps required to sample the first vertex (when sampled from Markov chain). To increase the number of samples per query, notice that if we sample $B = [v_1, \dots, v_{k-1}]$ via lifting, we can get graphlets induced

Algorithm 2 Ordered Lift Estimator (with optional shotgun sampling)

input Graph G, k-graphlet H_m
output $\hat{N}_m(G)$

```
Count |co(H_m)|- the number of compatible orderings in H_m.
Initialize v at an arbitrary node, n ← 0, N̂_m(G) ← 0
while stopping criteria is not met do
    Sample v_1 from π_1(v)
    Initialize V_S ← {v_1} and E_S ← {}
    Initialize 𝒩_e(S) ← 𝒩_e(v_1)
    Initialize π(S) ← π_1(v_1)
    while |V_S| < k − 1 do
        Sample an edge e = (v, u) uniformly from 𝒩_e(S), with
        v ∈ V_S and u ∉ V_S
        Set E_S(u) ← {(v, u) ∈ 𝒩_e(S)}
        Update π(S) ← π(S) |E_S(u)| / Σ_{v∈V_S} deg(v)−2|E_S|
        Update V_S ← V_S ∪ {u} and E_S ← E_S ∪ E_S(u)
        Query 𝒩_e(u)
        Update 𝒩_e(S) ← [𝒩_e(S) ∪ 𝒩_e(u)] \ E_S(u)
    end while
    if not shotgun sampling then
        Sample an edge e = (v, u) uniformly from 𝒩_e(S), with
        v ∈ V_S and u ∉ V_S
        Set E_S(u) ← {(v, u) ∈ 𝒩_e(S)}
        Set π(T) ← π(S) |E_S(u)| / Σ_{v∈V_S} deg(v)−2|E_S|
        Set V_T ← V_S ∪ {u} and E_T ← E_S ∪ E_S(u)
        if (V_T, E_T) ~ H_m then
            Update N̂_m(G) ← N̂_m(G) + π^{-1}(T)
        end if
    end if
    if shotgun sampling then
        for all u ∈ 𝒩_v(S) do
            Set E_S(u) ← {(v, u) ∈ 𝒩_e(S)}
            Set V_T ← V_S ∪ {u} and E_T ← E_S ∪ E_S(u)
            if (V_T, E_T) ~ H_m then
                Update N̂_m(G) ← N̂_m(G) + π^{-1}(S)
            end if
        end for
    end if
    Update n ← n + 1
end while
Normalize N̂_m(G) ← 1/n · 1/|co(H_m)| · N̂_m(G)
```

by $A = [v_1, \dots, v_{k-1}, u]$ for all $u \in \mathcal{N}_v(B)$ without using any additional queries:

$$N_m(G) = \frac{1}{|\mathrm{co}(H_m)|} \sum_A \mathbb{1}(G|A \sim H_m) =$$

$$\frac{1}{|\mathrm{co}(H_m)|} \sum_B \sum_{u \in \mathcal{N}_v(B)} \frac{\mathbb{1}\left(G|(B+u) \sim H_m\right)}{\tilde{\pi}_{k-1}(B)} \tilde{\pi}_{k-1}(B) =$$

$$= \frac{1}{|\mathrm{co}(H_m)|} \mathbb{E}_{\tilde{\pi}_{k-1}} \frac{\sum_{u \in \mathcal{N}_v(B)} \mathbb{1}\left(G|(B+u) \sim H_m\right)}{\tilde{\pi}_{k-1}(B)}$$

Let $B_1, \dots, B_n$ be sequences sampled from V_G^{k-1} via lifting process. Then

$$\frac{1}{n} \frac{1}{|\mathrm{co}(H_m)|} \sum_{i=1}^{n} \frac{\sum_{u \in \mathcal{N}_v(B_i)} \mathbb{1}\left(G|(B_i+u) \sim H_m\right)}{\tilde{\pi}_{k-1}(B_i)} \approx N_m(G),$$

where we only make queries to the vertices of B_i. We call this technique **shotgun sampling**.

Shotgun sampling is useful in cases when the number of queries is limited, but has a drawback of high correlation between samples. If the initial vertex v_1 was sampled from a random walk, the number of steps between initial vertices should be chosen so that shotgun shots are sufficiently far apart from each other.

3.3 Unordered lift estimator

We can also use the estimator (**??**) for the sampling space of k-graphlets $\mathcal{V}_k(G)$. In order to do that, we need to compute the probability $\pi_k(T)$ of getting a particular k-graphlet $T \in \mathcal{V}_k(G)$ in the lifting procedure. This can be done either recursively or directly. Let the set of vertices of T be $V_T = \{v_1, \dots, v_k\}$.

Recursive formula starts with known probabilities of initial vertices $\pi_1(v_i)$. Given a k-graphlet T, the probability of getting T via lifting is given by the sum $\pi_k(T) = \sum_S \mathbb{P}(T|S)\pi_{k-1}(S)$, where the sum is taken over all $k-1$-subgraphlets $S \subset T$, and $\mathbb{P}(T|S)$ denotes the probability of getting from S to T in the lifting procedure. Let $T \setminus S$ be the unique vertex of T that is not in S. Then

$$\pi_k(T) = \sum_{S \subset T} \pi_{k-1}(S) \frac{\deg_S(T \setminus S)}{|\mathcal{N}_e(S)|} = \sum_{S \subset T} \pi_{k-1}(S) \frac{|E_T| - |E_S|}{\sum_{u \in S} \deg(u) - 2|E_S|},$$

where the sum is taken over all $k-1$-subgraphlets $S \subset T$.

For a direct formula, we notice that $\pi(T) = \sum_{A \in \mathrm{co}(T)} \tilde{\pi}(A)$, where $\mathrm{co}(T)$ is the set of compatible orderings of T from previous section, and $\tilde{\pi}(A)$ is the probability of getting sequence $A \in \mathrm{co}(T)$ in the lifting process (see (**??**),(**??**)).

Then

$$\pi_k(T) = \sum_{A \in \mathrm{co}(T)} \frac{f(\deg(A[1]))}{K} \prod_{r=1}^{k-1} \frac{|E_{S_{r+1}(A)}| - |E_{S_r(A)}|}{\sum_{i=1}^{r} \deg(A[i]) - 2|E_{S_r(A)}|} \tag{4}$$

Although calculation of that probability on-the-fly is cost-prohibitive, we can greatly reduce the number of operations by noticing that the probability $\pi_k(T)$ is a function of degrees of the vertices: for a graphlet T of type m, let $[v_1, \dots, v_k]$ be the natural labelling of vertices in T induced by the isomorphism $H_m \to T$ with $d_i = \deg(v_i)$, then the probability of T is

$$\pi_k(T) = \frac{1}{K} F_m(d_1, \dots, d_k)$$

for some cached function F_m.

Example. Consider a triangle, which is a 3-graphlet with edges (v_1, v_2), (v_2, v_3) and (v_1, v_3). Given the degrees d_1, d_2, d_3

of the corresponding vertices, the probability function is

$$\pi_3(\text{triangle}) = \left(\frac{\pi_1(d_1)}{d_1} + \frac{\pi_1(d_2)}{d_2}\right)\frac{2}{d_1+d_2-2} + \\ + \left(\frac{\pi_1(d_2)}{d_2} + \frac{\pi_1(d_3)}{d_3}\right)\frac{2}{d_2+d_3-2} + \\ \left(\frac{\pi_1(d_3)}{d_3} + \frac{\pi_1(d_1)}{d_1}\right)\frac{2}{d_3+d_1-2}. \quad (5)$$

Example. Consider a wedge, which is a 3-graphlet with edges (v_1, v_2) and (v_1, v_3). Given the degrees d_1, d_2, d_3 of the corresponding vertices, the probability function is

$$\pi_3(\text{wedge}) = \left(\frac{\pi_1(d_1)}{d_1} + \frac{\pi_1(d_2)}{d_2}\right)\frac{1}{d_1+d_2-2} + \\ \left(\frac{\pi_1(d_1)}{d_1} + \frac{\pi_1(d_3)}{d_3}\right)\frac{1}{d_1+d_3-2}. \quad (6)$$

We need to only compute functions F_m once before starting the algorithm. When a k-graphlet T is sampled via lifting procedure, we find the natural labelling of vertices in T via the isomorphism $H_m \to T$, and use the function F_m together with the degrees $d_1, \dots, d_k$ of vertices of T to compute the value of $\pi_k(T) = \frac{1}{K}F_m(d_1, \dots, d_k)$.

Recursive formula (**??**) is useful for analysis of graphlet counts $N_m(G)$ for all types m, and direct formula (**??**) is useful for analysis of a specific graphlet type m.

Algorithm 3 Unordered Lift Estimator

input Graph G, k-graphlet H_m
output $\hat{N}_m(G)$

```
Set an ordering [1, ..., k] on the vertices of H_m, precompute
the function F_m(d_1, ..., d_k) and the global constant K
Initialize v at an arbitrary node, n ← 0, N̂_m(G) ← 0
while stopping criteria is not met do
    Sample initial vertex v from π_1(v)
    Initialize V_T ← {v} and E_T ← {}
    Initialize N_e(T) ← N_e(v)
    while |V_T| < k do
        Sample an edge e = (v, u) uniformly from N_e(T), with
        v ∈ V_T and u ∉ V_T
        Set E_T(u) ← {(v, u) ∈ N_e(T)}
        Update V_T ← V_T ∪ {u} and E_T ← E_T ∪ E_T(u)
        Query N_e(u)
        Update N_e(T) ← [N_e(T) ∪ N_e(u)] \ E_T(u)
    end while
    if (V_T, E_T) ~ H_m then
        Determine the ordering [v_1, ..., v_k] of vertices in V_T
        induced by the isomorphism (V_T, E_T) ~ H_m
        Set d_i = |N_e(v_i)| for all i = 1, ..., k
        Set π(T) = (1/K) F_m(d_1, ..., d_k)
        Update N̂_m(G) ← N̂_m(G) + π^{-1}(T)
    end if
    Update n ← n + 1
end while
Normalize N̂_m(G) ← (1/n) N̂_m(G)
```

3.4 Sampling a starting vertex

Framework 1. It is no problem to sample the starting vertex v independently and from an arbitrary distribution π_1 when we have access to all the vertices. Thus we sample starting vertices $v_1, \dots, v_n$ independently, and apply the lifting process to get independent graphlet samples $T_1, \dots, T_n$. The variance of the estimator $\hat{N}_m(G)$ is

$$\text{Var}(\hat{N}_m(G)) = \frac{1}{n}\text{Var}_{\pi_k}\left(\frac{\mathbb{1}(T \sim H_m)}{\pi_k(T)}\right) = \\ \frac{1}{n}\left(\sum_T \frac{\mathbb{1}(T \sim H_m)}{\pi_k(T)} - N_m(G)^2\right), \quad (7)$$

which is small when the distribution of $\frac{1}{\pi_k(T)}$ is close to uniform on k-graphlets T of type m.

The variation in (**??**) can be reduced by an appropriate choice of π_1, i.e. the starting distribution.

Example. For $k = 3$, let $\pi_1(v) = \frac{1}{K}\deg(v)(\deg(v)-1)$, where $K = \sum_{u \in V_G} \deg(u)(\deg(u)-1)$. Then by (**??**) and (**??**)

$$\pi_3(\text{triangle}) = \frac{6}{K}, \quad \pi_3(\text{wedge}) = \frac{2}{K}.$$

Calculating K takes $O(|V_G|)$ operations (preparation), sampling starting vertex v takes $O(\log(|V_G|))$ operations, and lifting takes $O(\Delta)$, where Δ is the maximum vertex degree in G.

Framework 2. When we don't have access to the whole graph structure, the best choice of sampling a starting vertex is via a simple random walk with transitional probabilities $p(i \to j) = \frac{1}{\deg(i)}$ whenever j in connected to i with an edge. Then stationary distribution

$$\pi_1(v) = \frac{\deg(v)}{2|E_G|},$$

and we can calculate all probabilities π_k accordingly.

Interesting feature of this choice of π_1 is that edge distribution π_2 is uniform: $\pi_2(e) = \frac{1}{|E_G|}$ for all $e \in E_G$. Therefore, this method is equivalent to sampling an edge uniformly at random and start lifting procedure from that edge.

More importantly, in order to get two ε-uncorrelated samples T_i and T_{i+1}, we only need to sample two ε-uncorrelated starting vertices.

Definition 1. Given a random walk with transitional probabilities $\mathbb{P}(i \to j)$ and stationary distribution $\pi(i)$ on graph G, the mixing time is defined as

$$\tau_G(\varepsilon) = \min\left\{t \mid \sum_{j \in G}\left|\mathbb{P}^{(t)}(i \to j) - \pi(j)\right| < \varepsilon \quad \forall i \in G\right\}.$$

Mixing time of the simple walk on G can be bounded by the following formula from [**?**]:

$$\frac{\mu}{2(1-\mu)\log(\frac{1}{2\varepsilon})} \le \tau_G(\varepsilon) \le \frac{\log(n) + \log(\frac{1}{\varepsilon})}{1-\mu},$$

where μ is the second largest (by absolute value) eigenvalue of the adjacency matrix of G, and $n = |V_G|$

Taking $\kappa = (1-\mu)^{-1}$, we'll use the mixing time $\tau_G(\varepsilon) \approx \kappa(\log(n) + \log(\varepsilon^{-1}))$ as a number of steps required to sample a starting vertex. In turn, that will generate two ε-uncorrelated graphlet samples (see section ??). Note that κ can vary from 20 to 500 for different social graphs (for more information about mixing time in social graphs, see [**?**]).

On the other hand, it has been shown in [**?**] that the mixing time of SRW can be as large as $\Omega(n^{k-2})$, even when the graph G has mixing time of constant order. The lifting procedure allows us to mix samples on the level of vertices instead of the level of subgraphs, making the mixing much faster.

There is a choice of mixing steps τ_{mix} that we need to make. This choice would depend on the cost of queries: if queries are cheap, we'll do $\tau_{mix} \approx \kappa \log(n)$, and if the queries are expensive or limited, we choose some constant τ_{mix}. Further discussion on the choice of τ_{mix} when the number of queries is limited will be based on empirical results.

4 THEORETICAL ANALYSIS FOR LIFTING MCMC

We hypothesized that the lifted MCMC will have faster mixing times than the graphlet MCMC. The lifting is performed after a sample from the underlying graph MCMC, so in order for the lifting procedure to

4.1 Variance Control

Consider the expectation in the variance under independence term of (**??**):

$$\mathbb{E}_{\pi_k} \frac{\mathbb{1}(T \sim H_m)}{\pi_k(T)^2} = \sum_T \frac{\mathbb{1}(T \sim H_m)}{\pi_k(T)} \leq \frac{N_m(G)}{\min_T \pi_k(T)}. \quad (8)$$

In case when $\pi_v(v) = \frac{\deg(v)}{2|E_G|}$, we can rewrite $\pi_k(T)$ from (**??**) as

$$\pi_k(T) = \frac{1}{2|E_G|} \sum_{A \in \mathrm{co}(T)} \prod_{r=2}^{k-1} \frac{|E_{S_{r+1}(A)}| - |E_{S_r(A)}|}{\sum_{i=1}^r \deg(A[i]) - 2|E_{S_r(A)}|}$$

So we can estimate $\min_T \pi_k(T)$ by

$$\min_T \pi_k(T) \geq \frac{|\mathrm{co}(T)|}{2|E_G|} \min_{A=[v_1,\ldots,v_k]} \prod_{r=2}^{k-1} \frac{1}{\sum_{i=1}^r \deg(v_i)}$$

Denote the first k highest degrees of vertices in G as $\Delta_1, \ldots, \Delta_k$ and denote $D = \prod_{r=2}^{k-1}(\Delta_1 + \ldots + \Delta_r)$. Then

$$\mathbb{E}_{\pi_k} \frac{\mathbb{1}(T \sim H_m)}{\pi_k(T)^2} \leq N_m(G) \frac{2|E_G|}{|\mathrm{co}(H_m)|} D. \quad (9)$$

4.2 Mixing time of lifted MCMC and covariance control

Definition 2. Define the mixing coefficient of a stationary Markov chain with discrete state space $X_t \in \mathcal{X}$ as

$$\gamma_X(h) = \frac{1}{2} \max_{x_1 \in \mathcal{X}} \sum_{x_2 \in \mathcal{X}} |\mathbb{P}(X_{t+h} = x_2, X_t = x_1) - \pi(x_1)\pi(x_2)|, \quad (10)$$

where $\pi(x)$ is the stationary distribution of the Markov chain.

Definition 3. Define the mixing time of a stationary Markov chain $\{X_t\}$ as

$$\tau_X(\varepsilon) = \min\{\, h \mid \gamma_X(h) < \varepsilon\} \quad (11)$$

THEOREM 1. *[**?**] Given stationary Markov chain $\{X_t\}$ with $\mu < 1$ being the second largest eigenvalue of the transitional matrix,*

$$\gamma_X(h) \leq e^{-(1-\mu)h}. \quad (12)$$

We assume that the starting vertex is sampled from the simple Markov process $X = \{X_t\}$ on V_G. We want to pick a time step τ so that the starting vertex v_k is sampled from $X_{k\tau}$, and graphlet T_k is then obtained from v_k by lifting. For large k, vertices v_k have almost stationary distribution π_v.

Define function $\phi(T) = \frac{\mathbb{1}(T \sim H_m)}{\pi_k(T)}$ that acts on the set $\mathcal{V}_k(G)$ of k-graphlets. Let $T_1, \ldots, T_n$ be the sequence of graphlets sampled via lifting. Given two starting vertices v_i and v_j, notice that random variables $T_i|v_i$ and $T_j|v_j$ are independent.

Pick t large enough so that the distribution of v_t is close to π_v. Given large n, the covariance term from (**??**) can be approximated as follows

$$\frac{2}{n^2} \sum_{i<j} \mathrm{Cov}_{\pi_k}\left(\phi(T_i), \phi(T_j)\right) \approx \frac{2}{n}\Big(\mathrm{Cov}_{\pi_k}(\phi(T_t), \phi(T_{t+1})) + \mathrm{Cov}_{\pi_k}(\phi(T_t), \phi(T_{t+2})) + \ldots\Big).$$

Consider

$$\mathbb{E}_{\pi_k}(\phi(T_t)\phi(T_{t+h})) = \mathbb{E}_{\pi(v_t)\times\pi(v_{t+h})}\mathbb{E}_{\pi_k}(\phi(T_t)\phi(T_{t+h})|v_t, v_{t+h}) = \mathbb{E}_{\pi(v_t)\times\pi(v_{t+h})}\left(\mathbb{E}_{\pi_k}(\phi(T_t)|v_t)\,\mathbb{E}_{\pi_k}(\phi(T_{t+h})|v_{t+h})\right).$$

Thus

$$|\mathrm{Cov}_{\pi_k}(\phi(T_t), \phi(T_{t+h}))| \leq \sum_{x_1, x_2 \in V_G} \mathbb{E}_{\pi_k}(\phi(T_t)|v_t = x_1)\,\mathbb{E}_{\pi_k}(\phi(T_{t+h})|v_{t+h} = x_2) \cdot |\mathbb{P}(v_t = x_1, v_{t+h} = x_2) - \pi_v(x_1)\pi_v(x_2)| \leq \max_{x_2 \in V_G} \mathbb{E}_{\pi_k}(\phi(T_{t+h})|v_{t+h} = x_2) \sum_{x_1} \mathbb{E}_{\pi_k}(\phi(T_t)|v_t = x_1) \cdot \sum_{x_2} |\mathbb{P}(v_t = x_1, v_{t+h} = x_2) - \pi_v(x_1)\pi_v(x_2)| \leq 2\gamma_X(h\tau) \max_{x_2 \in V_G} \mathbb{E}_{\pi_k}(\phi(T_{t+h})|v_{t+h} = x_2) \cdot \sum_{x_1} \mathbb{E}_{\pi_k}(\phi(T_t)|v_t = x_1).$$

Since

$$\sum_{x_1} \mathbb{E}_{\pi_k}(\phi(T_t)|v_t = x_1) \leq \max_{x_1} \frac{1}{\pi_v(x_1)} \sum_{x_1} \mathbb{E}_{\pi_k}(\phi(T_t)|v_t = x_1)\,\pi_v(x_1) \leq 2|E_G|N_m(G), \quad (13)$$

and, using notation $D = \prod_{r=2}^{k-1}(\Delta_1 + \ldots + \Delta_r)$,

$$\max_{x \in V_G} \mathbb{E}_{\pi_k}(\phi(T_t)|v_t = x) =$$

$$\max_x \sum_T \mathbb{1}(x \in V_T)\mathbb{1}(T \sim H_m)\frac{\mathbb{P}(T|x)}{\pi_k(T)} \leq$$

$$\max_x \frac{1}{\pi_v(x)}|\{T \mid x \in V_T\}| \leq 2|E_G|D.$$

Then

$$\frac{2}{n^2}\left|\sum_{i<j}\mathrm{Cov}_{\pi_k}\left(\phi(T_i), \phi(T_j)\right)\right| \leq$$

$$\frac{16}{n}N_m(G)|E_G|^2 D\sum_{h=1}^{\infty}\gamma_X(h\tau) \leq$$

$$\frac{16}{n}N_m(G)|E_G|^2 D\frac{e^{-(1-\mu)\tau}}{1-e^{-(1-\mu)\tau}}.$$

4.3 Mixing Time

In order to assess the mixing time of the we will use the concept of β-mixing.

Definition 4. The β-mixing coefficient of a Markov chain with discrete state space, $X_t \in \mathcal{X}$, is

$$\beta_X(t,h) :=$$

$$\sum_{x_0, x_1 \in \mathcal{X}} |\mathbb{P}\{X_t = x_0, X_{t+h} = x_1\} - \mathbb{P}\{X_t = x_0\}\mathbb{P}\{X_{t+h} = x_1\}|.$$

THEOREM 2. *Let X_t be the random walk over the graph G and let Y_t be the graphlet extracted from the k-lifting process. Then*

$$\beta_X(t,h) \geq \beta_Y(t,h). \tag{14}$$

PROOF. Let $Q_1(x_t, x_{t+h}) = \mathbb{P}\{X_t = x_t, X_{t+h} = x_{t+h}\}$ and $Q_2 = \mathbb{P}\{X_t = x_t\}\mathbb{P}\{X_{t+h} = x_{t+h}\}$. Also, let P_1, P_2 be the joint distributions of Y_t, Y_{t+h} when X_t, X_{t+h} are drawn according to Q_1, Q_2 respectively. Due to the lifting process, the distribution of $Y_t, Y_{t+h}|X_t, X_{t+h}$ is invariant to the choice of distribution of X_t, X_{t+h} (being from Q_1 or Q_2). Hence, by the generalized data processing inequality we have that

$$D_f(Q_1||Q_2) \geq D_f(P_1||P_2),$$

for any f-divergence. By selecting the total variation divergence, we have the result. □

5 EXPERIMENTS

5.1 Simulations

5.2 Network data

6 CONCLUSION

Acknowledgments

Grant support, people that helped.