Graph 4-Pattern Counting[1]

Disclaimer: these notes contain errors (w.h.p); if you find any, please let me know!

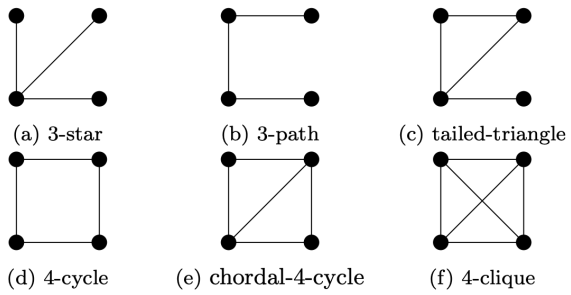
These notes present a fast, yet simple algorithm for sampling the number different of 4-vertex connected subgraphs in a large network. The algorithm is due to Jha, Sesahdhri and Pinar (2014).

- The study of small subgraphs is ubiquitous in network science. Understanding frequencies of subgraphs– for example, cliques (complete subgraphs)– provide us with insights into the nature of interdependency of a network.
- Let's start off with a couple of definitions that we will use throughout, which are so central to the problem that I'll state them before even defining the problem.

  Definition. A (connected) n**-motif** is a (connected) subgraph on n vertices.

  Definition. An **induced, connected 4-motif** (which I will henceforth just refer to as a **4-pattern**) is a connected 4-vertex subgraph which occurs without removing any edges between the vertices.

  Here[2] are all the all the possible 4-patterns, up to permutation of vertices:



- With these definitions, we can state our problem quite easily.

  Problem. Given a graph G, what is the frequency of each type of 4-pattern in the graph?

- Some setup first. Define C_i to be the count of the ith 4-pattern (in lex-order on the image order above). Specifically, C_1 is the count of 3-stars, C_2 is the count of 3-paths, etc. Let's also define N_i to be the count of the ith 4-pattern, if we relax the constraint that the pattern must be induced. This definition, though not immediately useful, will prove to be important!

[1] These notes were written by Karun Ram (email); last modified 29th May, 2023

[2] Image taken directly from Jha, Sesahdhri and Pinar (2014)

- It's useful to specify a graph access model under which the algorithm operates.
  - We assume the graph is stored in a hash-table adjacency list, to allow for fast lookup times; this does introduce some randomization into the runtime of the algorithm, but w.h.p. our lookups occur in $O(1)$ time.
  - We assume we can access the degree of each vertex v, and a random neighbor of v, in $O(1)$ time.
  - The paper assumes one ability in constant time that is not guaranteed by the graph access model they specify, but it is not an unreasonable ask– accessing a random neighbor of u that is not v. This can be achieved by rejection sampling in *expected constant time*, or by making some other clever assumptions on the access model.
  - Note that under this model, we can sample an edge under any probability distribution from the graph in $O(1)$ time, using the degrees to perform weighted sampling.
- The naive algorithm is to check every 4-vertex subgraph, which does return the true counts, but is extremely slow. Even in the best case, where we're able to determine the pattern type in $O(1)$ time, the naive algorithm would still need to sample all $\binom{n}{4}$ subgraphs, giving an algorithm that runs in $\Theta(n^4)$ time. As always, though, sampling comes to the rescue! We're going to construct an unbiased estimator for each C_i.
- If we could sample connected 4-subgraphs at random, it would suffice to check the 4-pattern each sample induces, and we could easily construct unbiased estimator for the true counts. The only issue is that the naive approach to this sampling would sample vertices at random, check connectivity, and then proceed, which is likely to produce many extraneous sets of vertices. But the idea is not too far off from this, actually!
- A crucial observation (and the main contribution this paper makes) is that except for the 3-star, every 4-pattern contains a 3-path (a path on 4 vertices of length 3). So, we're first going to construct estimators for C_2, C_3, C_4, C_5, C_6 (we'll see later how to use these to estimate C_1) by sampling 3-paths. Each 3-path contains 4 vertices, and each set of 4 vertices induces exactly one 4-pattern. The reason this works is because each 4-pattern (except the 3-star) contains a fixed number of 3-paths.
- The first step is to design a sampler that samples a 3-path, such that any 3-path is sampled with uniform probability. For each edge $(u, v) \in E(G)$, let $w_{(u,v)} = (d_u - 1)(d_v - 1)$, where $d_x = \deg_G(x)$, and let $W = \sum_{(u,v)\in E(G)} w_{(u,v)}$. These values can be preprocessed, so our sampler doesn't need to recompute them each time.

Consider the following sampling procedure:

procedure PREPROCESSSAMPLE(G)
  $w \leftarrow$ empty dictionary of weights
  $W \leftarrow 0$
  for $(u, v) \in E(G)$ **do**
    $w[(u, v)] \leftarrow (d_u - 1)(d_v - 1)$
    $W \leftarrow W + w[(u, v)]$
  return w, W

---

procedure SAMPLE(G)
  sample $(u, v) \in_R E(G)$ s.t. $\mathbf{Pr}\left[(u, v) \text{ is sampled}\right] = \frac{w[(u,v)]}{W}$
  sample $u' \in_R$ neighbors of u other than v, u.a.r
  sample $v' \in_R$ neighbors of v other than u, u.a.r
  return $\{u', u, v, v'\}$

Notice that SAMPLE can return a triangle (if $u' = v'$) or a 3-path.

Lemma 1. Fix some 3-path $(u', u), (u, v), (v, v')$, with $u' \neq v'$. The probability that SAMPLE outputs this 3-path is $\frac{1}{W}$.

Proof. Denote by $\mathcal{E}$ the event that SAMPLE outputs $(u', u), (u, v), (v, v')$. Then,

$$\begin{aligned}
\mathbf{Pr}\left[\mathcal{E}\right] &= \mathbf{Pr}\left[u', v' \text{ are selected} | (u, v) \text{ is selected}\right] \cdot \mathbf{Pr}\left[(u, v) \text{ is selected}\right] \\
&= \frac{1}{(d_u - 1)(d_v - 1)} \cdot \frac{w_{(u,v)}}{W} \\
&= \frac{1}{W} \qquad (\text{since } w_{(u,v)} = (d_u - 1)(d_v - 1)).
\end{aligned}$$

□

- Now, what is the probability we sample a specific 4-pattern (remember, this is an induced subgraph by definition) in the graph? More precisely, our sampler returns 3 edges to us, so we consider the set of those vertices, and consider the 4-pattern they induce. What is the chance we sample that specific 4-pattern?

  To answer this question, we observe that a specific instance of the ith 4-pattern contains τ_i distinct 3-paths, where τ_i is the ith element in

  $$\tau := [0, 1, 2, 4, 6, 12].$$

  For example, an instance of a 4-clique ($i = 6$) contains twelve 3-paths, and an

instance of a tailed-triangle ($i = 3$) contains two 3-paths[3]. So, we sample a specific instance of a 4-pattern (of type i) with probability $\frac{\tau_i}{W}$.

- One final concern, before we write out the full algorithm: How will we estimate C_1? There's a really clever trick, building off the interdependency between the C_i values and the N_i values (remember these?). There's a really simple linear relationship between the C_i values and the N_i values, given by the following:

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 2 & 4 \\ 0 & 1 & 2 & 4 & 6 & 12 \\ 0 & 0 & 1 & 0 & 4 & 12 \\ 0 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix} = \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{pmatrix}.$$

  Though not immediately obvious, the values in the matrix follow pretty simply using the same techniques we used to define τ earlier (in fact, notice that the second row of the matrix is just τ!). Again, drawing out the motifs and looking at all the subgraphs they contain is instructive in seeing why this holds[4].

- What we care about is the linear equation for N_1: $N_1 = C_1 + C_3 + 2C_5 + 4C_6$. This is useful because we can compute N_1 easily: any vertex $v \in V(G)$ is the center of $\binom{d_v}{3}$ (non-induced) 3-stars, so $N_1 = \sum_{v \in V(G)} \binom{d_v}{3}$. Thus, if we have unbiased estimators for C_3, C_5, C_6, we have an unbiased estimator for C_1:

$$\widehat{C_1} = N_1 - \widehat{C_3} - 2\widehat{C_5} - 4\widehat{C_6}. \qquad (*)$$

- Finally, the algorithm!

```
1:  procedure 3-PATH-SAMPLER(G, k: number of samples)
2:      PREPROCESSSAMPLE(G)
3:      v ← [0, 1, 2, 4, 6, 12]
4:      count ← [0, 0, 0, 0, 0, 0]
5:      for 1 ≤ j ≤ k do
6:          S ← SAMPLE(G)
7:          if |S| = 4 then          ▷ If |S| ≠ 4, then SAMPLE returns a triangle
8:              i ← subgraph type induced by S in G
9:              count[i] ← count[i] + 1
10:         else
```

[3] To see these, try drawing out all possible 3-paths they contain

[4] I don't think there's any better way to understand this; I couldn't see where it comes from at first either.

11: **continue** ▷ We reject if we sample a triangle.
12: **for** $2 \leq i \leq 6$ **do**
13: $\widehat{C_i} \leftarrow (count[i]/k) \cdot (W/v[i])$
14: $\widehat{C_1} \leftarrow \sum_{v \in V(G)} \binom{d_v}{3} - \widehat{C_3} - 2\widehat{C_5} - 4\widehat{C_6}$
15: **return** $\{\widehat{C_i} : i \in [1..6]\}$

- It remains to prove that the $\widehat{C_i}$s are unbiased estimators, and that when $k = \lceil \frac{1}{4\epsilon^2} \ln \frac{2\sqrt{e}}{\delta} \rceil$[5], each $\widehat{C_i}$ is a "good" estimator for the corresponding C_i.

Theorem 2. For any $i \in [1..6]$, $\mathbf{Exp}[\widehat{C_i}] = C_i$.

Proof. First we show the result for $i \in [2..6]$. For $i, j \in [2..6] \times [1..k]$, let X_{ij} be an indicator variable, such that

$$X_{ij} = \begin{cases} 1 & \text{In the } j\text{th iteration of 3-PATH-SAMPLER, } S \text{ induces the } i\text{th pattern} \\ 0 & \text{otherwise.} \end{cases}$$

By earlier remarks we can see that $\mathbf{Exp}[X_{ij}] = C_i \cdot \frac{\tau_i}{W}$ (since we sample each instance of a 4-pattern of type i with probability $\frac{\tau_i}{W}$, and there are C_i such instances). Then

$$\begin{aligned} \mathbf{Exp}[\widehat{C_i}] &= \frac{W}{k \cdot v[i]} \mathbf{Exp}[count[i]] \\ &= \frac{W}{k \cdot v[i]} \left[\sum_{j=1}^{k} \mathbf{Exp}[X_{ij}] \right] \\ &= \frac{W}{k \cdot v[i]} \cdot k \left[C_i \cdot \frac{v[i]}{W} \right] = C_i. \end{aligned}$$

When $i = 1$, by $(*)$, $\mathbf{Exp}[C_1] = \mathbf{Exp}[N_1 - \widehat{C_3} - 2\widehat{C_5} - 4\widehat{C_6}] = N_1 - C_3 - 2C_5 - 4C_6 = C_1$. □

Before we prove the next theorem, we state a useful bound, that is similar to the Chernoff bound, but (arguably) more powerful. The proof is omitted[6], but here is the result.

[5] The authors use a slightly tighter bound, but I cannot get the computations to match up with theirs, so I present this one which is much easier to show.

[6] See here for a complete proof.

Theorem 3 (Hoeffding)**.** Let $X_1, X_2, ..., X_\ell$ be independent random variables over support $[0,1]$. Let $\overline{X} = \frac{1}{\ell}\sum_{i=1}^{\ell} X_i$, and let $\mu = \mathbf{Exp}[\overline{X}]$. Then for $\epsilon \in (0,1)$ we have that

$$\mathbf{Pr}\left[|\overline{X} - \mu| \geq \epsilon\right] \leq 2\exp(-2k\epsilon^2).$$

Now the next theorem becomes easy to prove.

Theorem 4. Let $k = \lceil \frac{1}{4\epsilon^2} \ln \frac{2\sqrt{e}}{\delta} \rceil$. Then 3-PATH-SAMPLER returns estimators $\widehat{C}_i$ such that

$$\mathbf{Pr}\left[|\widehat{C}_i - C_i| \geq \frac{\epsilon W}{\tau'[i]}\right] \leq \delta,$$

where $\tau'[i] = [1, 1, 2, 4, 6, 12]$. In particular, $\mathbf{Pr}\left[|\widehat{C}_i - C_i| \geq \epsilon W\right] \leq \delta$.

Proof. Define X_{ij} as in Theorem 2 for $i \in [2,6]$. Let $Y_i = \frac{1}{k}\sum_{j=1}^{k} X_{ij}$. For fixed i, the X_{ij} values are iid. Further, $\mathbf{Exp}[Y_i] = C_i \cdot \frac{\tau'[i]}{W}$. Since $|\widehat{C}_i - C_i| \geq \frac{\epsilon W}{\tau'[i]}$ iff $|Y_i - \mathbf{Exp}[Y_i]| \geq \epsilon$, it follows that

$$\begin{aligned}
\mathbf{Pr}\left[|\widehat{C}_i - C_i| \geq \frac{\epsilon W}{\tau'[i]}\right] &= \mathbf{Pr}\left[|Y_i - \mathbf{Exp}[Y_i]| \geq \epsilon\right] \\
&\leq 2\exp\left(-2k\epsilon^2\right) \\
&\leq 2\exp\left(-2\frac{1}{4\epsilon^2}\ln\left(\frac{2\sqrt{e}}{\delta}\right)\epsilon^2\right) \\
&= 2\exp\left(-\frac{1}{2}\ln\left(\frac{2\sqrt{e}}{\delta}\right)\right) \\
&= \frac{1}{\sqrt{e}}\delta\exp(\frac{1}{2}) = \delta.
\end{aligned}$$

This gives the result for $i \in [2..6]$. When $i = 1$, we might be concerned about errors adding up. But we can get the same bound with some careful algebra.

$$X_{1,j} = \begin{cases} \frac{1}{2} & \text{In the } j\text{th iteration of 3-PATH-SAMPLER, } S \text{ induces a tailed-triangle} \\ \frac{2}{6} & \text{In the } j\text{th iteration of 3-PATH-SAMPLER, } S \text{ induces a chordal 4-cycle} \\ \frac{4}{12} & \text{In the } j\text{th iteration of 3-PATH-SAMPLER, } S \text{ induces a 4-clique} \end{cases}$$

□

Again let $Y_1 = \frac{1}{k}\sum_{j=1}^{k} X_{1k}$. Then $\mathbf{Exp}[Y_1] = (C_3 + 2C_5 + 4C_6)/W$. But $\widehat{C_1} = N_1 - WY_1$, so it follows that

$$\begin{aligned}\mathbf{Pr}\left[|C_1 - \mathbf{Exp}[C_i]| \geq \epsilon\frac{W}{\tau'[1]}\right] &= \mathbf{Pr}\left[|C_1 - \mathbf{Exp}[C_i]| \geq \epsilon W\right] \\ &= \mathbf{Pr}\left[|Y_1 - \mathbf{Exp}[Y_1]| \geq \epsilon\right] \\ &\leq \delta \qquad \text{(by the same method as the previous case).}\end{aligned}$$

So the theorem holds for any $i \in [1..6]$.

- The only thing left to do is bound the running time. By simple inspection (under our graph access model) it is clear that the algorithm runs in time $O(m+k)$: preprocessing takes time $O(m)$, and each sample runs in time $O(1)$, giving an overall runtime of $O(m+k)$.
- And we're done! This algorithm is quite simple, but powerful. There's a way to get a slight improvement to the parameter k, but the details are a bit too subtle for these notes.
- An natural question is whether we can generalize this approach: can we extend this beyond 4-patterns? Intuitively, to generalize this method, one would need to exhibit a similar linear relationship between induced and non-induced subgraph counts, and identify common patterns in 5-motifs (or larger!). The challenge here is that there are far more 5-motifs than 4-motifs, and far more 6-motifs than 5-motifs, so this task becomes (exponentially) more difficult. But perhaps there is a clever trick[7] that allows us to do this! We could also restrict our search to a simpler set of motifs; perhaps, those containing 4-paths, or beyond. To paraphrase: this is an open question! :)

Learning Tidbits:

- **Sample smart!** *Here, by sampling by 3-path instead of by vertex, we get 4-patterns much more frequently, and the estimators are much more simple to compute.*
- **Exploit variable relationships!** *We used the other variables to avoid sampling for $\widehat{C_1}$ separately, which is convenient given our method of sampling wouldn't pick it up.*

[7] To the best of my knowledge, nobody has found this yet.