Streaming Triangle Counting: Deep Dive

A Step-by-Step Visualization of the $O(m\kappa/T)$ Algorithm

Intuition Behind Sampling with $\frac{\min(d(u),d(v))}{d_E}$

The key idea is:

"Sample edges in proportion to how likely they are to participate in triangles."

Algorithm Recap

We aim to estimate the number of triangles T using $O(\frac{m\kappa}{T})$ space over three passes. The passes are:

1. Edge sampling proportional to degree.
2. Sampling a random neighbor of the lower-degree endpoint.
3. Checking triangle closure.

Pass 1: Degree-Proportional Edge Sampling

Objective

Select $N = \Theta(d_E/T)$ edges, each with probability $d(e)/d_E$, where $d_e := \min(d_u, d_v)$. This biases toward edges in dense regions.

1. Normalization: Turning Weights into Probabilities

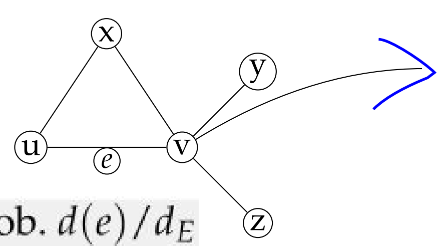
- **Problem**: Raw $\min(d(u), d(v))$ values are just *weights*—they don't sum to 1 (so they're not valid probabilities).
- **Solution**: Divide each edge's weight by the total weight d_E.

$$\mathbb{P}(\text{sample edge } e) = \frac{\min(d(u), d(v))}{d_E}$$

This ensures:

- Each edge's sampling probability is between 0 and 1.
- All probabilities sum to 1 (i.e., we always sample *some* edge).

Visualization



Sample e with prob. $d(e)/d_E$

Details

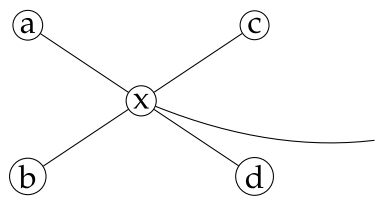
- Compute in a stream the sum $d_E = \sum_e d(e)$ via a degree oracle.
- Use weighted reservoir sampling to pick N edges proportional to $d(e)$.

Pass 2: Random Neighbor of Lower-Degree Endpoint

Objective

For each sampled edge $e = (u, v)$, find the endpoint $x = \arg\min\{d(u), d(v)\}$. Query a uniformly random neighbor w of x.

Visualization



Pick random neighbor w

2. Probability of Picking the "Closing" Neighbor

After sampling $e = (u, v)$:

- Let $x = \arg\min(d(u), d(v))$ (the lower-degree endpoint)
- w must be the third vertex in Δ
- Probability of randomly selecting w from x's neighbors:

$$\mathbb{P}(\text{pick } w \mid \text{sampled } e) = \frac{1}{\min(d(u), d(v))}$$

(Uniform random selection from x's neighbors)

Details

- Use degree oracle to identify x with smaller degree.
- Use neighbor oracle to sample w uniformly from $N(x)$.

3. Joint Probability per Edge

$$\mathbb{P}(\text{detect } \Delta \text{ via } e) = \underbrace{\frac{\min(d(u), d(v))}{d_E}}_{\text{sampling } e} \times \underbrace{\frac{1}{\min(d(u), d(v))}}_{\text{picking } w} = \frac{1}{d_E}$$

Pass 3: Triangle Verification

Objective

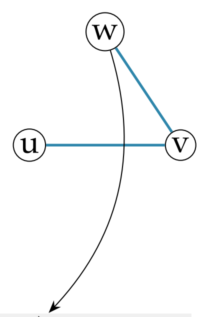
Check if w completes a triangle with u, v by verifying the existence of edge (u, w) or (v, w) in the stream.

Visualization

4. Total Detection Probability (All 3 Edges)

Since Δ can be detected via any of its 3 edges:

$$\mathbb{P}(\text{detect } \Delta) = 3 \times \frac{1}{d_E} = \frac{3}{d_E}$$



Check (u, w) in stream

Details

- Maintain a small hash or Bloom filter to record observed edges in second or third pass.
- If (u, w) observed, record success; else fail.

2. For *Any* Triangle (Across All T Triangles):

$$\mathbb{P}(\text{detect any triangle}) \approx \frac{3T}{d_E}$$

Probability and Space Analysis

Combining passes gives success probability per sample $3/d_E$, so $N = \Theta(d_E/T)$ samples suffice. Storing N edges and queries yields $O(m\kappa/T)$ space.

Created with LaTeX, Visualized for clarity

- **Why approximate?**
  - Strictly speaking, the events are not perfectly independent (edges may overlap), but for large graphs, this approximation holds well.