Homework 3: Mining Data Streams

Data Mining – 3

Lakshmi Srinidh Pachabotla

November 25, 2024

1. Introduction

This report presents the implementation and testing of a streaming graph processing algorithm for estimating transitivity using Reservoir Sampling. The algorithm is designed to be memory efficient, making it suitable for large streaming data sets. Two datasets were used for testing:

- Facebook (ego-Facebook): A large, sparse graph with 4039 nodes and 88,234 edges.
- Email (email-Eu-core): A smaller, denser graph with 1005 nodes and 16,706 edges.

The goal of this assignment was to evaluate the algorithms' ability to approximate graph transitivity while analyzing its behavior under wearing conditions including different reservoir sizes.

2. Methodology

- Algorithm Selection: Reservoir sampling was chosen for this assignment due to its ability to maintain a fixed-size random sample of edges from an unbounded edge stream. This sampling method ensures each edge in the stream has an equal probability of being included in the sample, regardless of its position in the stream.
- Reservoir Sampling Algorithm: The reservoir sampling algorithm processes edges sequentially and maintains a fixed-size reservoir of sampled edges. Each edge is either added to the reservoir (if it has available space) or replaces an existing edge based on a random probability.
- Wedge and Triangle counting:

- ✓ Wedges: A triplet of nodes connected by two edges.
- ✓ Triangles: A closed triplet where all three nodes are connected.
- ✓ Transitivity: It is estimated using the formula i.e., 3 multiplied by the Number of Triangles whole divided by the Number of Wedges.

3. Experiments

- Experimental Setup
  - ✓ Reservoir Sizes: 5000, 10000, 15000 and 20000 edges.
  - ✓ Iterations: Each reservoir size was tested over 5 iterations to account for random variability in sampling.
- Metrics
  - ✓ Estimated Transitivity: Calculated using sampled edges.
  - ✓ Exact Transitivity: Computed using NetworkX.
- Error Metrics
  - ✓ Absolute Error: |Estimated – Exact Transitivity|.
  - ✓ Relative Error: |Estimated – Exact Transitivity| * 100 whole divided by Exact Transitivity.

4. Results

- **Facebook Dataset**

Reservoir Size	**Estimated Transitivity**	**Exact Transitivity**	**Absolute Error**	**Relative Error (%)**
5000	0.086	0.519	0.433	83.34
10,000	0.179	0.519	0.340	65.39

15,000	0.262	0.519	0.257	49.41
20,000	0.348	0.519	0.171	32.97

- ✓ Even at 20,000 edges, the estimation transitivity remains far below the exact value of 0.519. This is due to the sparse triangle distribution in the graph. Random sampling struggles to include edges forming triangles, as the probability of sampling such edges decreases with sparsity. The results suggest that further increasing the reservoir size could improve accuracy.

- Email Dataset

Reservoir Size	**Estimated Transitivity**	**Exact Transitivity**	**Absolute Error**	**Relative Error (%)**
5000	0.301	0.267	0.034	12.76
10,000	0.531	0.267	0.264	98.80
15,000	0.773	0.267	0.506	189.14
20,000	0.851	0.267	0.584	218.49

- ✓ This demonstrates that smaller reservoirs work well for dense graphs. At larger reservoir sizes (for example 15,000 edges), the reservoir becomes less representative as it disproportionately includes wedges and triangles. This results in inflated transitivity estimates with relative error exceeding 200% at 20,000 edges. For estimation is due to the biased sampling introduced by larger reservoirs in dense graphs. Random sampling selects a high number of edges from densely connected regions, creating an unbalanced representation of the graph. At 20,000 edges the reservoir size exceeds the total edge count, effectively making it a complete subgraph of the original graph. This breaks the probabilistic nature of reservoir sampling, leading to extreme overestimation.

- Comparative Analysis
  - ✓ Sparse graphs. Increasing the reservoir size consistently reduces relative error. For dense graphs, larger reservoirs lead to diminishing returns and eventually degrade accuracy.
  - ✓ The strengths of reservoir sampling are that it is efficient for small dense graphs with manageable edge counts. And it provides reasonable approximations with smaller reservoirs for graphs with high triangle density.
  - ✓ The limitations of reservoir sampling are that it struggles to capture global structure in large sparse graphs. Bayas introduced by larger reservoirs lead to over-estimation in dense graphs. Fixed-size reservoirs cannot adapt to the increasing hedge count in unbounded streams.

5. Bonus Questions and Answers

- What were the challenges you faced?

For large sparse graphs example, Facebook sampled edges failed to capture sufficient triangles leading to under-estimation. In the same way, in dense graphs, for example, email, larger reservoirs disproportionately sampled wedges and triangles, causing overestimation. The algorithm required multiple iterations to stabilize results.

- Can the algorithm be easily parallelized?

Yes, reservoir sampling can be parallelized by dividing the edge stream into chunks and sampling independently. Merging results while maintaining the reservoir size. Dependencies in triangle counting across edges may increase communication overhead.

- Does the algorithm work for unbounded streams?

Yes, the algorithm works for unbounded streams as it maintains a fixed-size reservoir. However, there are some limitations to this as the accuracy may degrade as the stream grows indefinitely without refreshing the reservoir. Using time-decayed sampling to prioritize newer edges could be an improved method.

- Does the algorithm support edge deletions?

No, the current implementation does not support edge deletions.

6. Conclusion

Reservoir Sampling provides a memory-efficient method for estimating transitivity in streaming graphs. While the algorithm performs well for smaller, denser graphs, it struggles with sparse graphs and large reservoirs, leading to underestimation or overestimation.