Exact and Parallel Triangle Counting in Dynamic Graphs

Devavret Makkar, David A. Bader, Oded Green
Outline

Background
- Triangle Counting
- Dynamic Graphs

Implementation
- Graph updating
- Triangle updating

Evaluation
- Batch size
- Breakdown
- Speedup

Discussion or Questions
Triangle Counting
Background
Triangle Counting

Applications
- Finding transitivity
- Spam detection in email networks
- Finding tightly knit communities
- Finding trusses $k$-trusses
- Evaluating the quality of different community detection algorithms

$$T = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples of nodes in the network}}.$$
Background
Triangle Counting

Current Approaches

- Enumerating over all node triplets $O(V^3)$
- Using linear algebra operations
- Adjacency list intersection (using hash tables)
Dynamic Graphs
Background
Dynamic Graphs

Useful for larger graphs with evolving datasets

Needs two things

1) Dynamic data structure
2) Algorithm to update the metric of interest

Should be computationally inexpensive compared to restarting the computation from scratch

Should produce the same result as the static graph algorithm
Background
Dynamic Graphs

Existing dynamic graph frameworks

- STINGER (DISTINGER for distributed systems and cuSTINGER for GPUs)
- AIMS
- GraphIN

- Why Stinger?
  - More flexible than CSR
  - Supports update operations
  - Better locality than a linked list
  - Lower storage bound
Dynamic Graph Updating
Dynamic Graph Updating

Bunch multiple changes to a graph into ‘batches’

Given a batch update, create an update-graph (G’)

Represent the update-graph as a CSR and sort that update-graph

Assuming the original graph was and still is already sorted, merge G’ and G
Dynamic Graph Updating
Insertion

1: procedure **INSERTION**
2: parallel for $u \in V$ do
3:     $i \leftarrow d^G_u$  \quad \triangleright \text{degree of } u \text{ in } G
4:     $j \leftarrow d^{G'}_u$  \quad \triangleright \text{degree of } u \text{ in } G'
5: while $i \geq 0 \land j \geq 0$ do
6:     $\text{diff} \leftarrow \text{adj}(u, G)[i] - \text{adj}(u, G')[j]$
7:     if $\text{diff} > 0$ then
8:         $\text{adj}(u, G)[i + j + 1] \leftarrow \text{adj}(u, G')[i]$  \quad \triangleright \text{Copy from original.}
9:         $i \leftarrow i - 1$
10: else
11:     $\text{adj}(u, G')[i + j + 1] \leftarrow \text{adj}(u, G')[j]$  \quad \triangleright \text{Copy from batch graph.}
12:     $j \leftarrow j - 1$
13: end if
14: end while
15: while $j \geq 0$ do
16:     $\text{adj}(u, G')[i + j + 1] \leftarrow \text{adj}(u, G')[j]$
17:     $j \leftarrow j - 1$
18: end while
19: end procedure
Dynamic Graph Updating

Deletion

---

**procedure** DELETION

parallel for \( u \in V \) do

\[ i \leftarrow d^G_u \]
\[ j \leftarrow d^{G'}_u \]

while \( i \geq 0 \) and \( j \geq 0 \) do

\[ \text{diff} \leftarrow \text{adj}(u, G)[i] - \text{adj}(u, G')[j] \]

if \( \text{diff} = 0 \) then \( \text{adj}(u, G)[i] \leftarrow \text{NULL} \)

end if

if \( \text{diff} \geq 0 \) then \( i \leftarrow i - 1 \)

end if

if \( \text{diff} \leq 0 \) then \( j \leftarrow j - 1 \)

end if

end while

\[ i \leftarrow 0 \]
\[ j \leftarrow 0 \]

while \( i < d_u \) do

if \( \text{adj}(u, G)[i] \neq \text{NULL} \)

\[ \text{adj}(u, G)[i] \leftarrow \text{adj}(u, G)[j]; j \leftarrow j + 1 \]

end if

\[ i \leftarrow i + 1 \]

end while

\[ d_u \leftarrow j \]

end procedure
Triangle Count Updating
Triangle Counting

Types of Triangles

- $\Delta^1_i$ (triangles with 1 new edge and 2 old edges)
- $\Delta^2_i$ (triangles with 2 new edges and 1 old edge)
- $\Delta^3_i$ (triangles with 3 new edges)

$\text{NewTriangles} = |\Delta^1_i| + |\Delta^2_i| + |\Delta^3_i|$
Triangle Counting

Break up discovery by num new edges

For each edge $<u,v>$ in the batch update, intersect the adjacency lists

$$s_{e,1} = \text{adj}(u, \tilde{G}_I) \cap \text{adj}(v, \tilde{G}_I)$$

$$S_1^i = 2 \cdot |\Delta_1^i| + 4 \cdot |\Delta_2^i| + 6 \cdot |\Delta_3^i|$$

$$S_2^i = \sum_{e \in E'} |s_{e,2}| = 2 \cdot |\Delta_2^i| + 6 \cdot |\Delta_3^i|$$

$$S_3^i = 6 \cdot |\Delta_3^i|$$
Triangle Counting

After gathering and discover 1+, 2+ and 3+ new-edged triangles

Use Inclusion - Exclusion formula to compute total new triangles

\[
|\Delta_1^i| + |\Delta_2^i| + |\Delta_3^i| = \frac{1}{2} \left( S_1^i - S_2^i + \frac{S_3^i}{3} \right)
\]
Triangle Counting

Deleting edges is simpler

Look for edges that existed before their removal

Triangles do not get recounted

\[
S_1^d = 2 \cdot |\Delta_1^d| \quad (10)
\]

\[
S_2^d = 2 \cdot |\Delta_2^d| \quad (11)
\]

\[
S_3^d = 2 \cdot |\Delta_3^d| \quad (12)
\]

And from (10) + (11) + (12) we get

\[
|\Delta_1^d| + |\Delta_2^d| + |\Delta_3^d| = \frac{1}{2}(S_1^d + S_2^d + S_3^d) \quad (13)
\]
Triangle Counting

(a) 2
(b) 3
(c) 3
(d) 3
(e) 3
(f) 3
Evaluation
Evaluation
Networks Used

| Name                  | Network Type | $|V|$ | $|E|$ | Ref. | Static (sec.) | Insertion (sec) | Deletion (sec) |
|-----------------------|--------------|-----|-----|------|------|----------------|----------------|----------------|
| coPapersDBLP          | Social       | 540k| 30M | [3]  | 1.032| 0.053          | 0.452          | 0.025          |
| in-2004               | WebcrawI     | 1.38M| 27M | [3]  | 18.176| 0.213          | 2.208          | 0.117          |
| com-or-kut            | Social       | 3M  | 234M| [25] | 90.164| 0.242          | 1.107          | 0.218          |
| com-LiveJournal       | Social       | 4M  | 69M | [25] | 8.975 | 0.168          | 0.765          | 0.067          |
| cage15                | Matrix       | 5.15M| 94M | [3]  | 1.638 | 0.132          | 0.651          | 0.043          |
| nlpkkt160             | Matrix       | 8.3M | 221M| [3]  | 1.778 | 0.192          | 0.329          | 0.089          |
| road_central          | Road         | 14M | 33M | [3]  | 1.348 | 0.288          | 0.348          | 0.029          |
| nlpkkt200             | Matrix       | 16.2M| 432M| [3]  | 3.460 | 0.910          | 1.081          | 0.164          |
| road_usa              | Road         | 24M | 58M | [3]  | 2.188 | 0.480          | 0.550          | 0.046          |
Evaluation
Batch Size

(a) Insertions

(b) Deletions
Fig. 4. This figure depicts the execution breakdown (in percentage) for the three stages in the execution: 1) creating the update graph $G'$ from the batch update, 2) inserting (or deleting) the batches into the graph (modification of cuSTINGER), and 3) running the dynamic graph triangle counting.
Evaluation
Breakdown

Fig. 5. This figure depicts the execution breakdown (in percentage) of only the dynamic triangle counting analytic using the inclusion-exclusion formulation. For both the insertion (a) and deletions (b) there are three phases. The execution time of the triangle counting accounts for the purple bars in Fig. 4.
Evaluation

Speedup - compared to previous algorithm which recounted all triangles after each update
Questions?