THE INPUT/OUTPUT COMPLEXITY OF TRIANGLE ENUMERATION

Francesco Silvestri, University of Padova

Joint work with

Rasmus Pagh, IT University of Copenhagen

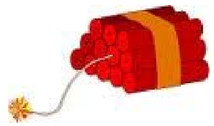
PODS 2014, Snowbird, Utah, US

Overview

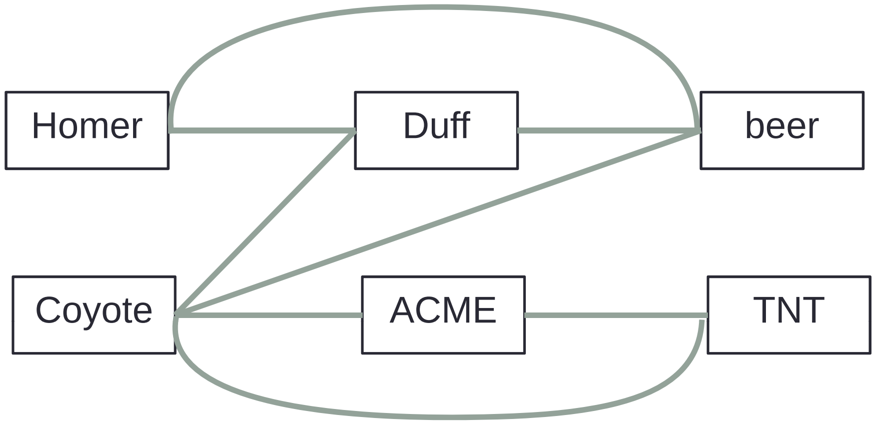
- Introduction
  - Triangles in graphs: Counting vs listing vs enumerating
  - Model of computation
  - Previous results
- Cache-aware algorithm
  - New randomized algorithm
  - Derandomization
- Cache-oblivious algorithm
  - Recursive approach
  - Randomized algorithm
- Lower bound
  - “Best-case” lower bound nearly matching the upper bounds

Sales person	Brand
H. Simpson	DUFF
W. Coyote	ACME
W. Coyote	DUFF

Product type	Brand
Beer	DUFF
TNT	ACME



Sales person	Product type
H. Simpson	Beer
W. Coyote	Beer
W. Coyote	TNT



Some problems with triangles

- **Counting triangles**
  - Compute the (approximate) number of triangles in a graph
  - Fast matrix multiplication, sampling…

- **Listing triangles**
  - **Generate and store** all triangles (write to external memory)

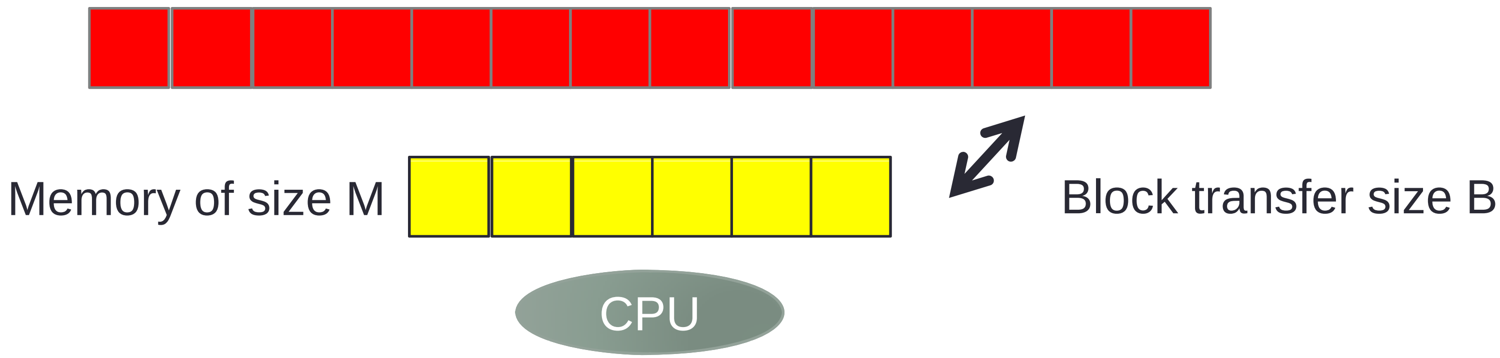
- **Enumerating triangles**
  - **Generate** all triangles in a graph
  - Do **not** store them (i.e., write them to external memory)

Enumerating vs listing

- Almost no difference in RAM model
  - Cost of generating a triangle = cost of storing a triangle
- Huge difference in the I/O model
  - Memory may not contain all T triangles: $\Omega(T/B)$ I/Os
  - Worst case $\Omega\left(E\sqrt{E}/B\right)$ I/Os (with $\sqrt{E}$-clique)
  - Larger than the cost of generating triangles $\Omega\left(E\sqrt{E}/\left(B\sqrt{M}\right)\right)$
- In many cases, we **don't need** to store the output
- Example in database systems
  - Pipeline operations may not require storing intermediate results

Memory hierarchy

- Input graphs are usually big; do not fit into internal memory.
- Use **I/O model** model [Vitter 2008] (external memory)
- Complexity of an algorithm: **number of I/Os**



- **Cache-oblivious algorithm**:
  Code does not use memory parameters M and B.

Previous work

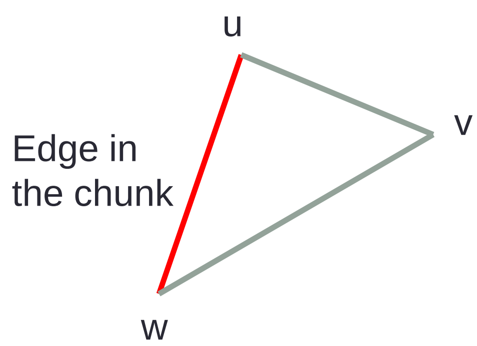
- All papers target the **listing** problem
- [Dementiev, PhD 2007] $O\left(\frac{E\sqrt{E}\log_{M/B}(E/B)}{B}\right)$
- [Menegola, TR 2010] $O\left(E+\frac{E\sqrt{E}}{B}\right)$
- [Hu, Tao, and Chung, SIGMOD 2013] $O\left(\frac{E^2}{BM}+\frac{T}{B}\right)$
  - Provide worst-case lower bound $\Omega\left(\frac{E\sqrt{E}}{B\sqrt{M}}+\frac{T}{B}\right)$

Do not exploit memory size M

Previous work: [Hu, Tao, Chung 2013]

1. Split edges into chunks of *M* edges
   - *O(E/M)* chunks

2. For each chunk:
   1. Load the chunk in memory
   2. Find all triangles with an edge in the chunk
      - Requires scanning the adjacency list of each vertex *v*



- Total I/O complexity: $O\left(\frac{E^2}{BM} + \frac{T}{B}\right)$

Our results

- I/O complexity given in **expectation**
- **Better** cache-aware algorithm using $O\left(\frac{E\sqrt{E}}{B\sqrt{M}}\right)$ $if\ M \geq \sqrt{E}$
- **Derandomization** of the cache-aware algorithm
- **Cache-oblivious** version with same complexity.
- **Best-case** lower bound of $\Omega\left(\frac{T}{B\sqrt{M}} + \frac{T^{2/3}}{B}\right)$

Some notation

- Vertices are ordered (e.g. by ID)
- Triangle represented by triplet (v_1, v_2, v_3) where $v_1 < v_2 < v_3$

- **Def**: A triangle (v_1, v_2, v_3) is **(c_1, c_2, c_3)-colored** if:
  - v_1 has color c_1, v_2 has color c_2, v_3 has color c_3

- **Def:** (c_1, c_2, c_3)-enumeration problem:
  - find all (c_1, c_2, c_3)-colored triangles

- Input: three edge sets

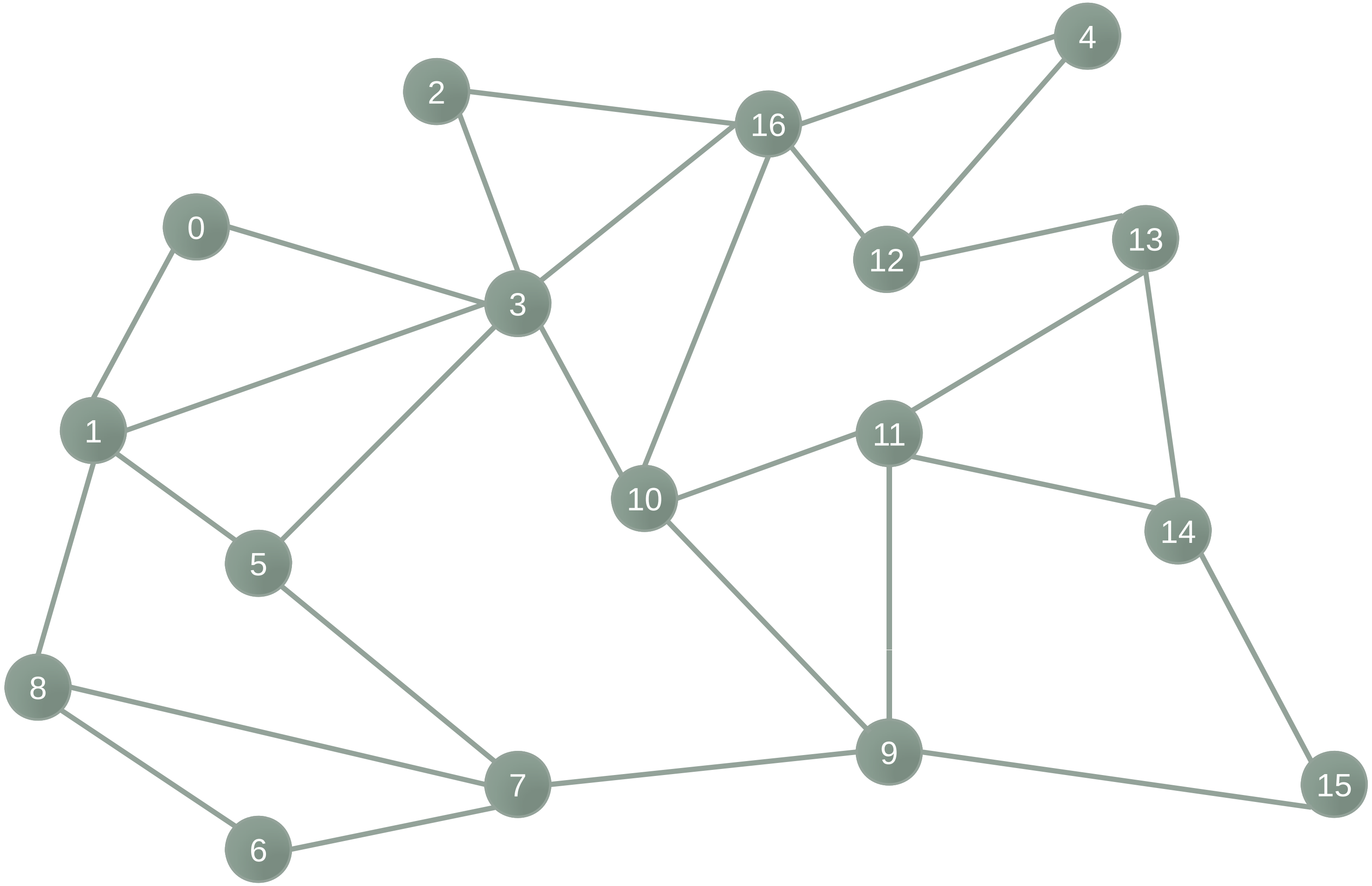
  $E_{c1, c2}$: edges with colors c_1, c_2 , $E_{c1, c3}$: edges with colors c_1, c_3, $E_{c2, c3}$: edges with colors c_2, c_3,

Cache-aware algorithm

1. Randomly color each vertex independently and uniformly with $c = \sqrt{E/M}$ colors
   - A triangle can be colored in c^3 ways
   - 4-wise independent hash function suffices

1. For each color triplet (c_1, c_2, c_3)
   1. Consider edge sets compatible with coloring: $E_{c1, c2}$, $E_{c1, c3}$, $E_{c2, c3}$.
   2. Solve the (c_1, c_2, c_3)-enumeration problem with the [Hu, Tao, Chung 2013] algorithm.

Three colors: RED, GREEN, BLUE



Looking for triangles (v_1, v_2, v_3) colors RED, GREEN, BLUE

I/O Complexity, intuition

- Number of subproblems $c^3 = \frac{E}{M}\sqrt{\frac{E}{M}}$
- For each (c_1, c_2, c_3), we need: $E_{c1,\, c2}$, $E_{c1,\, c3}$, $E_{c2,\, c3}$
- Expected subproblem size: $\mathbf{E}\left[E_{c_1,c_2} + E_{c_1,c_3} + E_{c_2,c_3}\right] = 3M$
- Expected I/O of a subproblem $O\left(\frac{\left(E_{c_1,c_2} + E_{c_1,c_3} + E_{c_2,c_3}\right)^2}{BM}\right) = O(M/B)$
- **Total expected I/O:** $O\left(\frac{E}{B}\sqrt{\frac{E}{M}}\right)$

Except if each vertex has degree $< \sqrt{EM}$

Does not hold!

High degree vertices

- Vertex *v* is **high degree** if $\deg(v) \geq \sqrt{EM}$
- At most $2\sqrt{\frac{E}{M}}$ high degree vertices
- $\Gamma(v)$: adjacency list of vertex v

Reporting triangles containing a high degree vertex:

1. Sort edges by **small** vertex
2. Remove edges where the small vertex is not in $\Gamma(v)$
3. Sort remaining edges by **large** vertex
4. Remove edges where the large vertex is not in $\Gamma(v)$
5. Each remaining edge makes a triangle with *v*

I/O Complexity

- High degree vertices:

$$O\left(\sqrt{\frac{E}{M}}\frac{E}{B}\log_{M/B}(E/B)\right) = O\left(\frac{E}{B}\sqrt{\frac{E}{M}}\right) \quad if \quad M \geq E^{1/2}$$

- Random coloring:

$$O\left(\frac{E}{B}\sqrt{\frac{E}{M}}\right)$$

- **Total optimal expected I/O complexity**: $O\left(\frac{E}{B}\sqrt{\frac{E}{M}}\right)$

Derandomization

- We use a small family of 4-wise independent functions [Alon et al., 1992]
- We fix the color of each vertex in *log (E/M)* iterations
  - One bit in each iteration
- In each iteration, we compute how well each function balances subproblems
  - According to some “cost” function
  - It can be proved that a “good” function exists in the family

Cache-oblivious algorithm: idea

- Problems:
  - To identify "large degree" vertices, need M and B.
  - The number of colors depends on M and B.

- Solution sketch for (c_1, c_2, c_3)-enumeration
  - Remove "extremely large degree" vertices incident to a constant fraction of the edges.
  - Randomly color vertexes using 2 colors.
  - Recurse on 8 coloring problems (each of about 1/4 size).

I/O lower bound

- **Assumption:** information on edges/vertices are indivisible
  - For enumerating a triangle we need all its edges in memory at the same time

- **Best-case lower bound**: applies to each input with T triangles, and every possible algorithm execution

$$\Omega\left(\frac{T}{B\sqrt{M}}\right) \text{ I/Os}$$

- **Hardest graph** ($\sqrt{E}$ -clique), $T=\Omega\left(E^{3/2}\right)$. We will show:

$$\Omega\left(\frac{E^{3/2}}{B\sqrt{M}}\right) \text{ I/Os}$$

Reorganizing I/Os in rounds

- **A:** execution of an algorithm enumerating *T* triangles with *M* memory

- **A'**: simulation of A on a memory of size *2M* so that
  - A' can be decomposed in rounds
  - Each round starts with *M/B* inputs and ends with *M/B* outputs
  - Same asymptotic I/O complexity as A

- Ideas:
  - Half of the memory simulates the memory used by A
  - Half of the memory is used as buffer for I/Os

I/O lower bound, cont.

- **How many triangles can we generate with *M* edges?**
- Answer:

$$O(M\sqrt{M})$$

- **Triangles reported in each round:**
  - I/Os: $2M/B$
  - New triangles $O\left(M\sqrt{M}\right)$
- I/O lower bound $\Omega\left(\frac{T}{B\sqrt{M}}\right)$

Conclusion

- Optimal (in expectation) enumeration of triangles in external memory
  - Aware and oblivious algorithms

- The algorithm can be generalized to the extraction of other subgraphs (e.g., *k*-cliques) and parallelized.

- Open problem: can we derive *output sensitive* algorithms in the I/O model?

Thank you!