Cache-oblivious algorithm

- A cache-oblivious algorithm is an algorithm that does not use memory parameters $M$ and $B$ (Frigg et al., 1999). Blocks are moved between the two memory levels by an automatic replacement policy (e.g., OPT, IRU).

The I/O model

- Input graphs are usually big and do not fit in internal memory.
- Use the I/O model [Vitter 2008] (also known as external memory model):
  - Consists of two memory levels.
  - A (potentially) unbounded slow memory, named external memory.
  - A fast memory of size $M$ (in words), named internal memory.
- Data are moved between the two levels in blocks of $B$ words.
- The CPU can only read and write data in the internal memory.
- Complexity of an algorithm: number of I/O operations.

Cache-aware algorithm

Preliminaries

- Vertices are ordered (e.g., by ID).
- A triangle $(v_1, v_2, v_3)$ is $(e_1, e_2, e_3)$-colored if:
  - $v_1 \neq v_2 \neq v_3$, $v_1 \neq v_3$.
  - $v_1$ has color $c_1$, $v_2$ has color $c_2$, $v_3$ has color $c_3$.
- $E_{c_1, c_2, c_3}$: edges with colors $c_1, c_2, c_3$.
- $E_{c_1, c_2, c_3}$: edges with colors $c_1, c_2, c_3$.

I/O complexity

- For each $(c_1, c_2, c_3)$, we need: $K_{c_1, c_2, c_3}$.
- Each set has expected size $M$.

Expected number of blocks: $\left(\frac{E_{c_1, c_2, c_3}}{M} \right)$.

Lower bound

- Assumption: information on edges/vertices is indivisible: that is, for enumerating a triangle we need all its edges in memory at the same time.
- Best-case lower bound: applies to each input to $T$ triangles and every possible algorithm execution.
- Main lower bound: the enumeration of $T$ distinct triangles requires $\left(\frac{E_{c_1, c_2, c_3}}{M} \right)$ I/Os.

Sketch of the proof

- Let $J$ be the simulation of an algorithm enumerating $T$ triangles with $M$ memory.
- Let $A$ be the simulation of $A$ on a memory of size $2M$ so that $A$ can be decomposed in rounds:
  - Each round starts with $M$ blocks and ends with $M$ reads.
  - Some asymptotic I/O complexity as $A$.
- The simulation relies on the following ideas:
  - Half of the memory simulates the memory used by $J$.
  - Half of the memory is used as buffer for I/Ds.
- In each round, there are $M$ edges in memory.

References