Breaking the Linear-Memory Barrier in Massively Parallel Computing

Fast MIS on Trees with $n^\epsilon$ Memory per Machine

Manuela Fischer, ETH Zurich
joint work with Sebastian Brandt and Jara Uitto
Model:

Sublinear-Memory MPC
Model: Sublinear-Memory MPC
large-scale data does not fit onto a single machine centralized algorithms not applicable

parallel computation framework

Massively Parallel Computation

Karloff, Suri, Vassilvitskii [SODA’10]

Goodrich, Sitchinava, Zhang [ISAAC’11]
Beame, Koutris, Suciu [PODS’14]
Andoni, Nikolov, Onak, Yaroslavtsev [STOC’14]
Beame, Koutris, Suciu [JACM’17]
Czumaj, Łacki, Madry, Mitrović, Onak, Sankowski [arXiv’17]
Massively Parallel Computation (MPC) Model

$M$ machines with $S$ words local memory

synchronous rounds consisting of

- **local computation**
  - in parallel at every machine
  - unbounded computational power

- **global communication**
  - all-to-all
  - for every machine:
    - sent messages $\ll S$
    - received messages $\ll S$

**complexity:** number of rounds
Massively Parallel Computing (MPC) Model

$M$ machines with $S$ words local memory

$n$ nodes, $m$ edges, maximum degree $\Delta$

Note: edge may appear on two machines!
Model: Sublinear-Memory MPC
Model:
Sublinear-Memory MPC
Parameter Choice for MPC

\( M \) machines with \( S \) words local memory and \( M \cdot S = \Theta(m + n) \)

Linear Memory \( S = \tilde{O}(n) \)

usual assumption for traditional MPC algorithms

single machine can see all the nodes

unrealistic for large-scale data!

- \( \tilde{O}(n) \) might be prohibitively large
- sparse graphs admit trivial solution

Algorithms have been stuck at this linear-memory barrier!
Fundamentally?
Breaking the Linear-Memory Barrier:

**Efficient Sublinear-Memory MPC Algorithms**

\[ S = O(n^\varepsilon) \] local memory
\[ M = O(m/n^\varepsilon) \] machines
\[ \text{poly log log } n \] rounds

**imposed locality:**
machines see only subset of nodes, regardless of sparsity of graph

**our approach to cope with locality:**
enhance **LOCAL algorithms** with **global communication**
- exponentially faster than LOCAL algorithms due to shortcuts
- polynomially less memory than traditional MPC algorithms
MPC Algorithm for MIS on Trees

Our Result:

randomized $O(\log^3 \log n)$-round MPC algorithm

with $S = O(n^\varepsilon)$ memory that w.h.p. computes MIS on trees.

Brandt, F., Uitto 2018
Algorithm Outline

1) Shattering
   break graph into small components
   i) Degree Reduction
   ii) LOCAL Shattering

2) Post-Shattering
   solve problem on remaining components
   i) Gathering of Components Distributed Union-Find
   ii) Local Computation
Algorithm Outline

1) **Shattering**
   break graph into small components
   i) **Degree Reduction**
   ii) **LOCAL Shattering**

2) **Post-Shattering**
   solve problem on remaining components
   i) **Gathering of Components** *Distributed Union-Find*
   ii) Local Computation
Polynomial Degree Reduction: **Subsample-and-Conquer**

**Subsample**
- subsample nodes independently

**Conquer**
- compute random MIS in subsampled graph
  - gather connected components
  - locally compute random 2-coloring
  - add a color class to MIS

Non-subsampled **high-degree node**
- w.h.p. has many subsampled neighbors
- thus w.h.p. has at least one MIS neighbor
- hence will be removed from the graph