Massively Parallel 2

Eva Rotenberg

Borůvka's algorithm.

- Let T be a minimum spanning tree in some graph G.
- A T-fragment is a connected subgraph of T.
- Idea: build T by iteratively concatenating fragments.
- Beginning: Each point is a fragment.
- Step: For each fragment X, let $e = (x, y)$ be the cheapest edge between X and $G \setminus X$. Use e, combine X with Y ($y \in Y$).
- Analysis:
  - Correct? The cheapest edge of a cut belongs to an MST.
  - How many steps? After i iterations, each fragment $\geq 2^i$ vertices.

Question: Can we use this idea to compute spanning trees in Congest?
Can we use this idea to compute spanning trees in parallel?

Massively parallel minimum spanning tree computation

- Graph G has N vertices and M edges,
- $S = \sqrt{N}$ (small), $P = \tilde{O}(M/S)$ (many).

Challenges with implementing Borůvka?

- Representing the state
- Implement one "step" in constant many rounds

Representing the state: Represent each fragment by, say, lowest ID node in fragment.

Machine storing edge uv should be able to find fragment of u and of v.

Note: many machines.

Two hints for parallel Borůvka

- Challenge: In one step, newly joining edges and their components may form a long chain.
  $X \to Y \to Z \to \dots$
  To avoid this, use randomisation:
    - Every fragment chooses a random colour (yellow, green)
    - A smallest edge is only ‘valid’ if it goes from yellow to green.
    - In each round, add only ‘valid’ edges.
    - Probability $1/4$ an edge is valid; slows down by a constant factor.
    - Now, the green fragment coordinates the merge with (possibly many) yellow fragments.
- Challenge: A fragment does not fit into one machine, and the number of edges it receives even less so.
    - Build aggregation trees: $\sqrt[4]{N}$-ary rooted trees;
    - Edges arrive at the leaves and are filtered towards the root.
    - Filtering: Only the smallest edge is relevant.

Graph sketching (sketching cuts, connectivity)

- $S = \tilde{O}(N)$, P_0 coordinates.
- Warm-up: an edge from a cut.
- Assume you have a graph G and a subset A of the vertices of G.
- Every vertex knows whether it itself is in A, and knows the names of its edges vu.
- Find an edge crossing the cut from A to not-A?
- What if the cut is one edge?Every vertex of A sends xor of their edges to P_0. Then P_0 xors those and gets name of edge.
- What if there are between $k/2$ and k edges crossing the cut?Use predefined hashing function to sample with probability $1/k$.
- Note: should be coordinated! Vertices u and v either both sample or not-sample uv
- Expect $1/2$ to 1 edge across the cut to be sampled. With constant probability, we have sampled exactly one edge across the cut.

Challenge: Did we succeed?

Graph sketching

Idea: if we know how many edges cross a cut, we can use coordinated sampling to find such an edge with constant probability.
Challenge: did we succeed?
Idea: Name of edge uv is u, v, R_{uv},
where R_{uv} is a random string of $\Theta(\log n)$ bits (say, $80 \log n$), each bit is 1 with probability $1/8$.
Then the number of 1-bits is highly concentrated around its expected value, (less than $14 \ln n$ w.h.p)
and because the 1s are so sparse, it is very likely that if we xor two $R_e \neq R_{e'}$ the result has many more 1s. (more than $14 \ln n$ w.h.p)
if we xor even more we get even closer to half of the bits being 1.
Details: exercise.

Sketching Connectivity

Setup: $S = \tilde{O}(N)$ and P_0 coordinates. Wish to find spanning tree.

We can detect cut of size ca. k (e.g. $k/2$ to k) with constant probability.

Repeat $\log n$ times to get high probability.

Repeat for $\log n$ guesses for k: 1, 2, 4, 8, 16, 32, etc.

- Borůvka? ($\log n$ rounds.)

Use cut-sketching to find an edge crossing from fragment to rest-of-graph.

Every vertex samples $\log n \cdot \log n \cdot \log n$ edges. (That is, $\log^4 n$ bits)

Send those to P_0, then P_0 can simulate entire Borůvka and get a spannning tree.