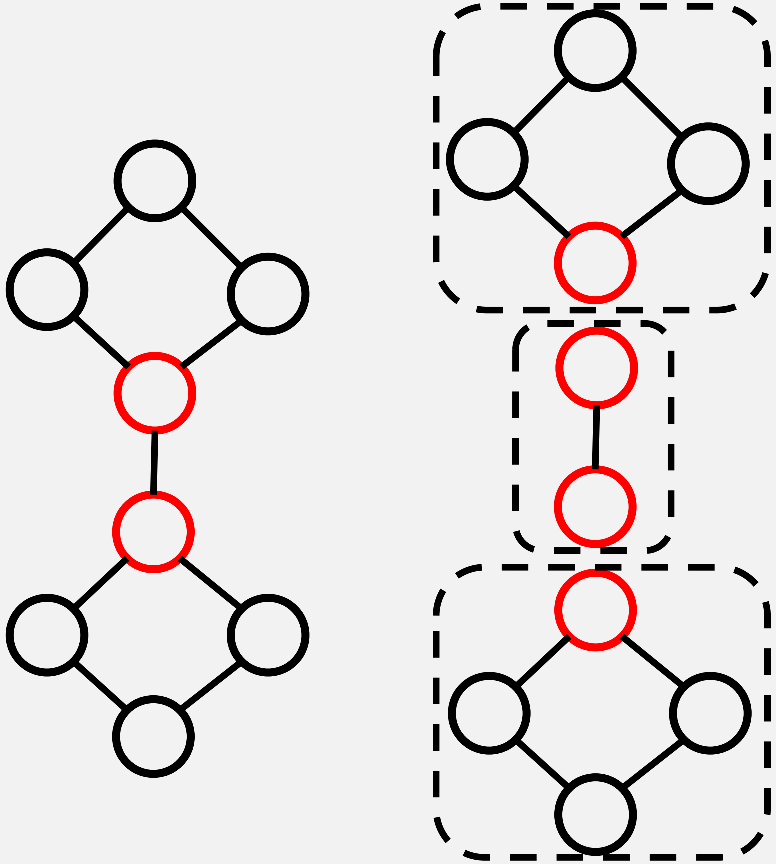
Distributed Algorithms for the Graph Biconnectivity and Least Common Ancestor Problems

Ian Bogle, George Slota

Rensselaer Polytechnic Institute

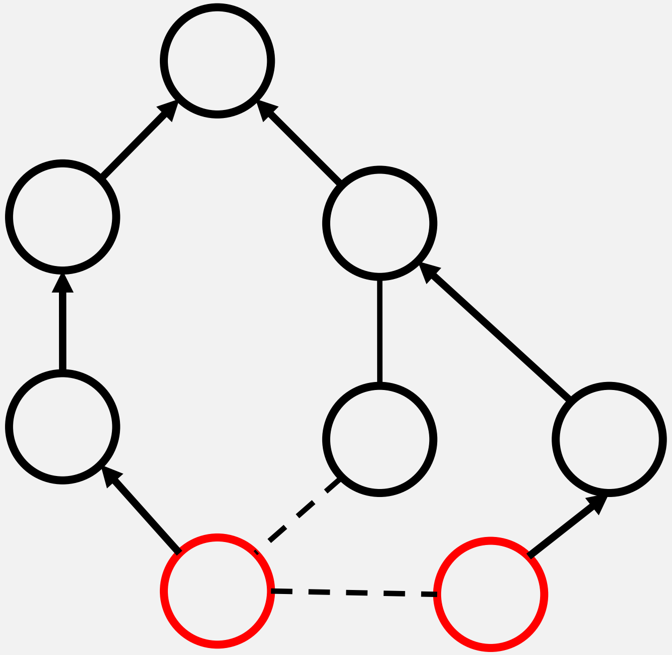
Graph Biconnectivity Finds Subgraphs That Remain Connected After One Vertex Is Removed

- Vertices that disconnect a graph are called Articulation Points, or Cut Vertices
- Biconnectivity algorithms find all articulation points
  - Either through complete edge labeling, or vertex labels
- Vertices can be in many biconnected components



Lowest Common Ancestors are Common Ancestors of Two Vertices in a BFS Tree

- Start at two vertices, traverse through parents until the first common ancestor (LCA) is reached
- Multiple LCA traversals are embarrassingly parallel
- Individual LCA traversals cannot be easily parallelized



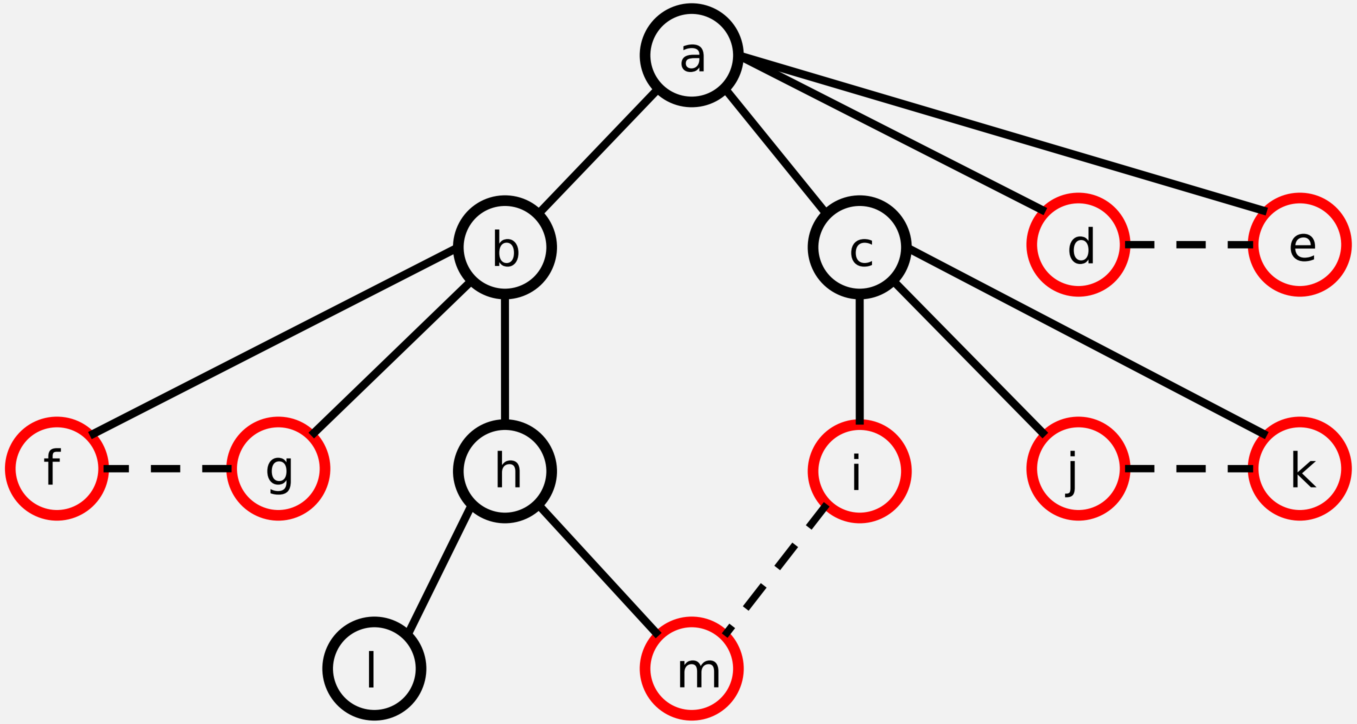
Biconnectivity Algorithms Have Been Widely Studied in Shared Memory

- Hopcroft and Tarjan 1973 – work optimal serial DFS-based approach
- Tarjan and Vishkin 1985 – Parallel approach based on building an auxiliary graph
- Slota and Madduri 2016 – Parallel approach using BFS and color propagation
- Chaitanya and Kothapalli 2016 – Parallel approach using the LCA Heuristic
- No efficient distributed memory approaches have been proposed, to our knowledge

BCC-LR intuition and general approach

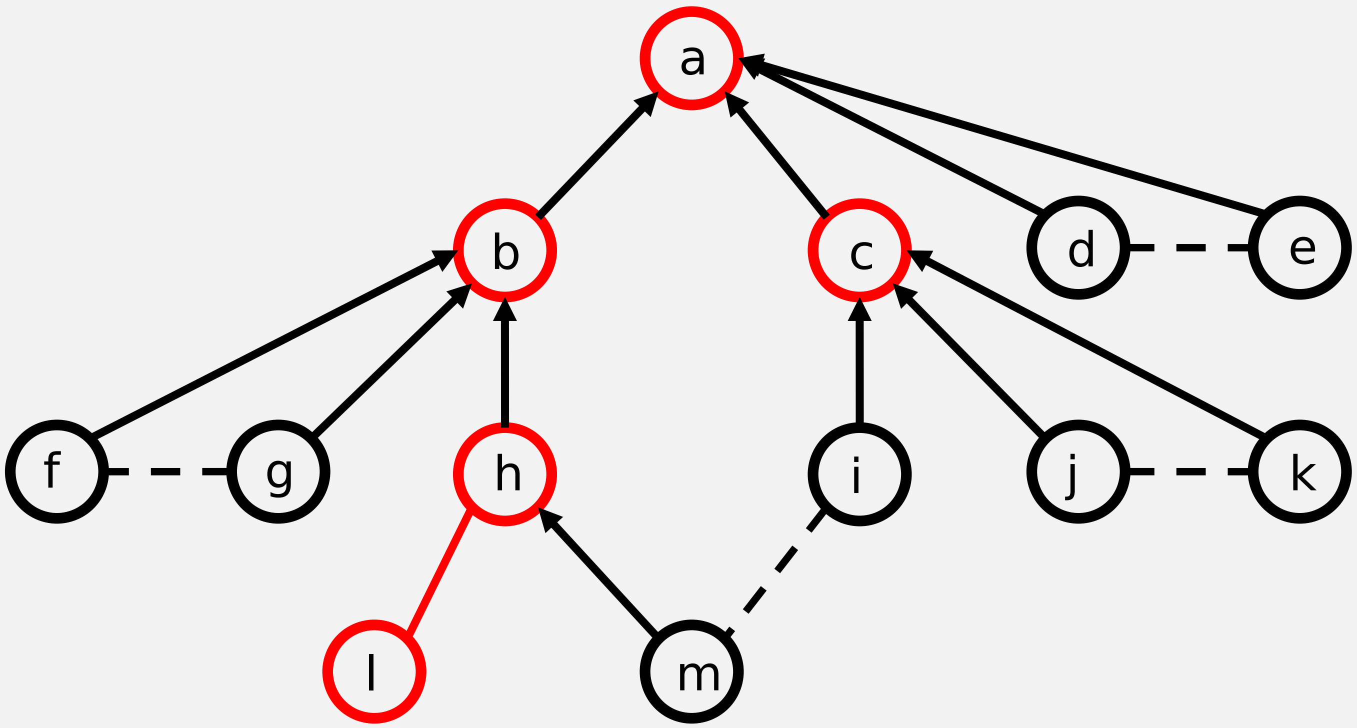
- Whitney’s 1932 theorem states that a graph is biconnected if there are two edge disjoint paths between each vertex.
- Our previous work used this idea to find degenerate features in ice sheet meshes
- Use LCA heuristic to identify potential articulation points
- Three label procedures to eliminate false positives:
  - Propagate LCA vertex IDs down the tree
  - Reduce LCA labels
  - Pass low labels to neighbors

LCA Heuristic Example



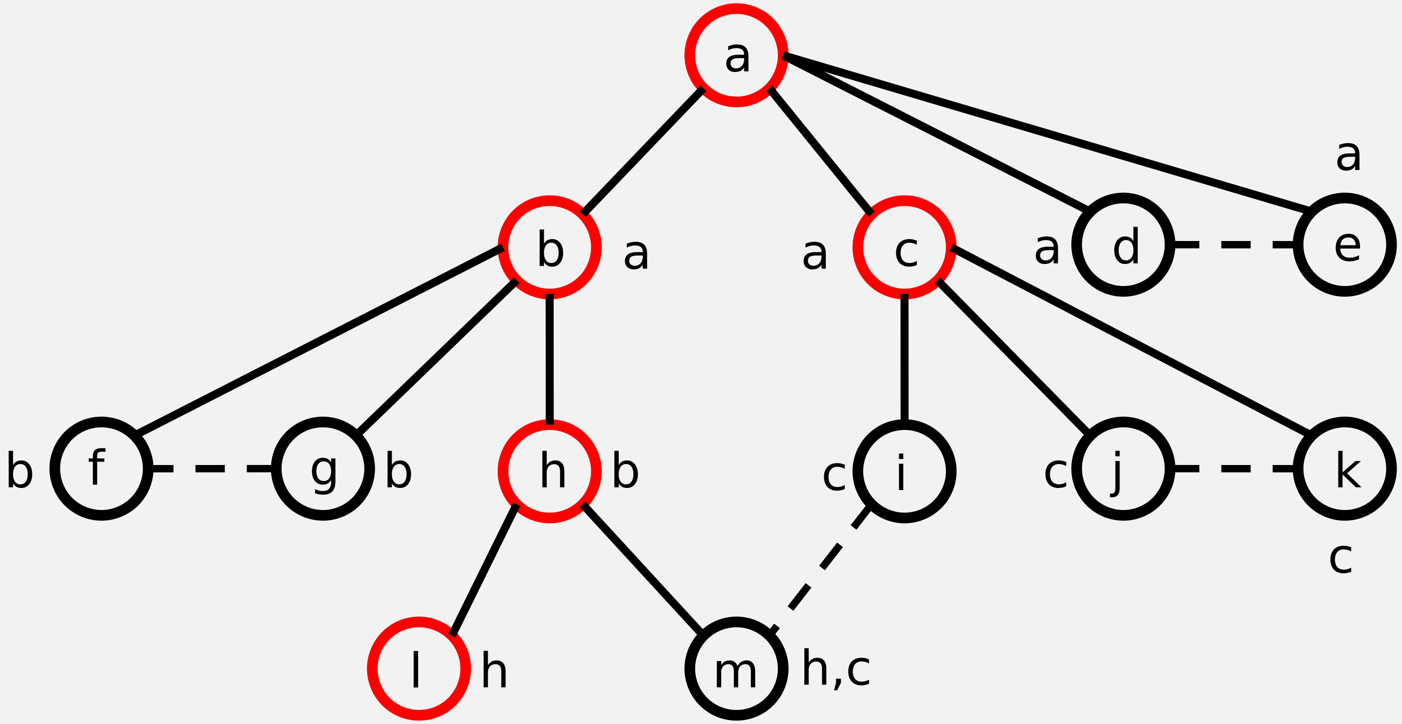
Find the LCAs of all non-tree edge endpoints

LCA Heuristic Example



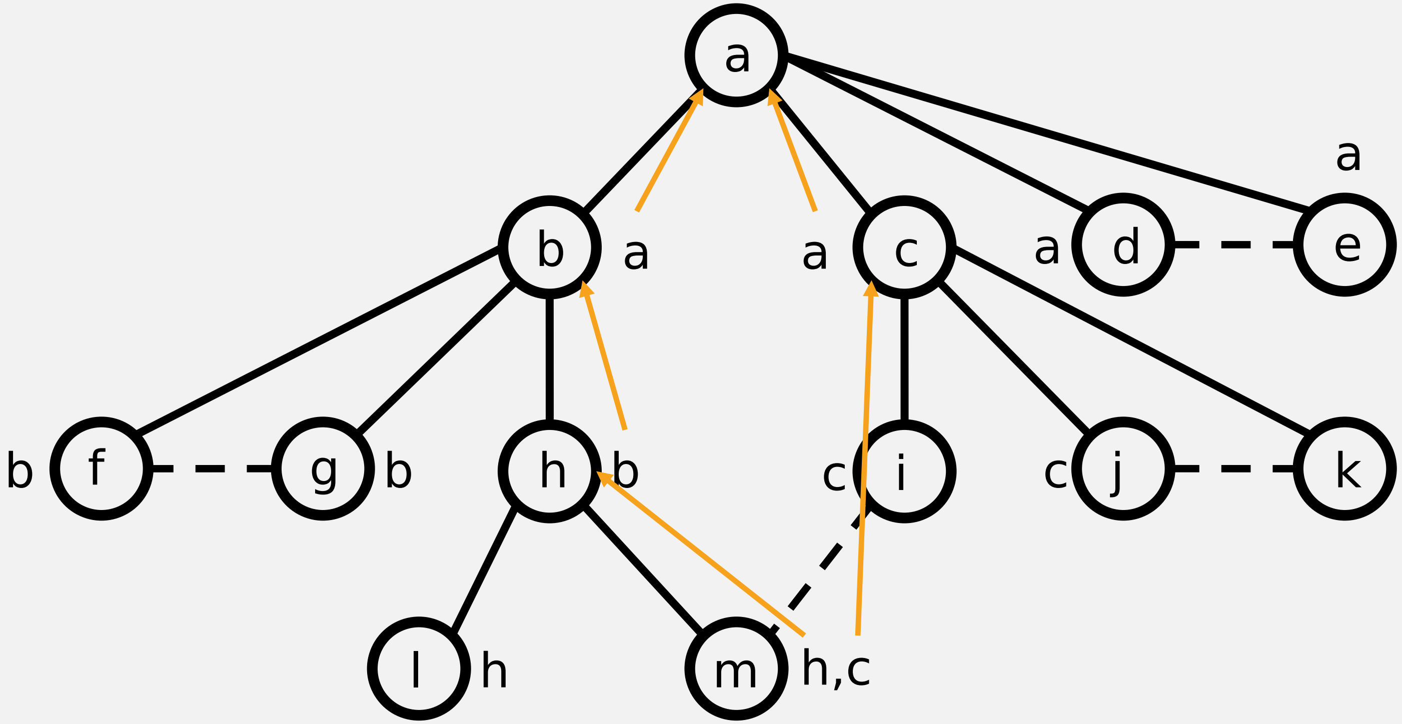
Flag all LCA vertices and endpoints of non-traversed edges as potential articulation points

BCC-LR Example



Label Propagation Step:
Propagate IDs of LCA vertices down the tree, allowing for multiple labels to accumulate when necessary

BCC-LR Example

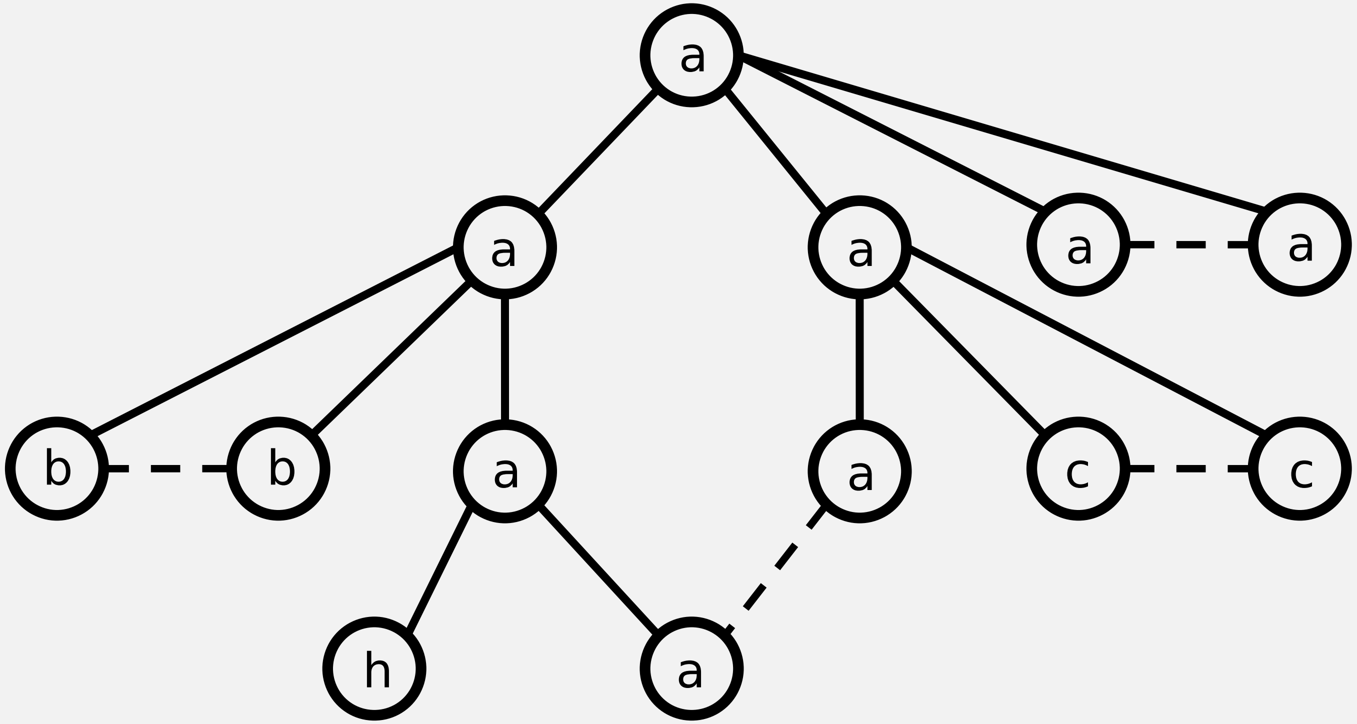


Label Reduction:
On vertices with multiple LCA IDs, replace the LCA with the highest level with its own label.

Once the vertex has only one LCA ID, pass that up the tree

Do an LCA traversal with h and c as start points, using labels instead of parents

BCC-LR Example



Low label propagation: LCA labels do not uniquely identify biconnected components, so we use low labels passed upwards to ensure they don’t travel through articulation points

Distributed memory considerations

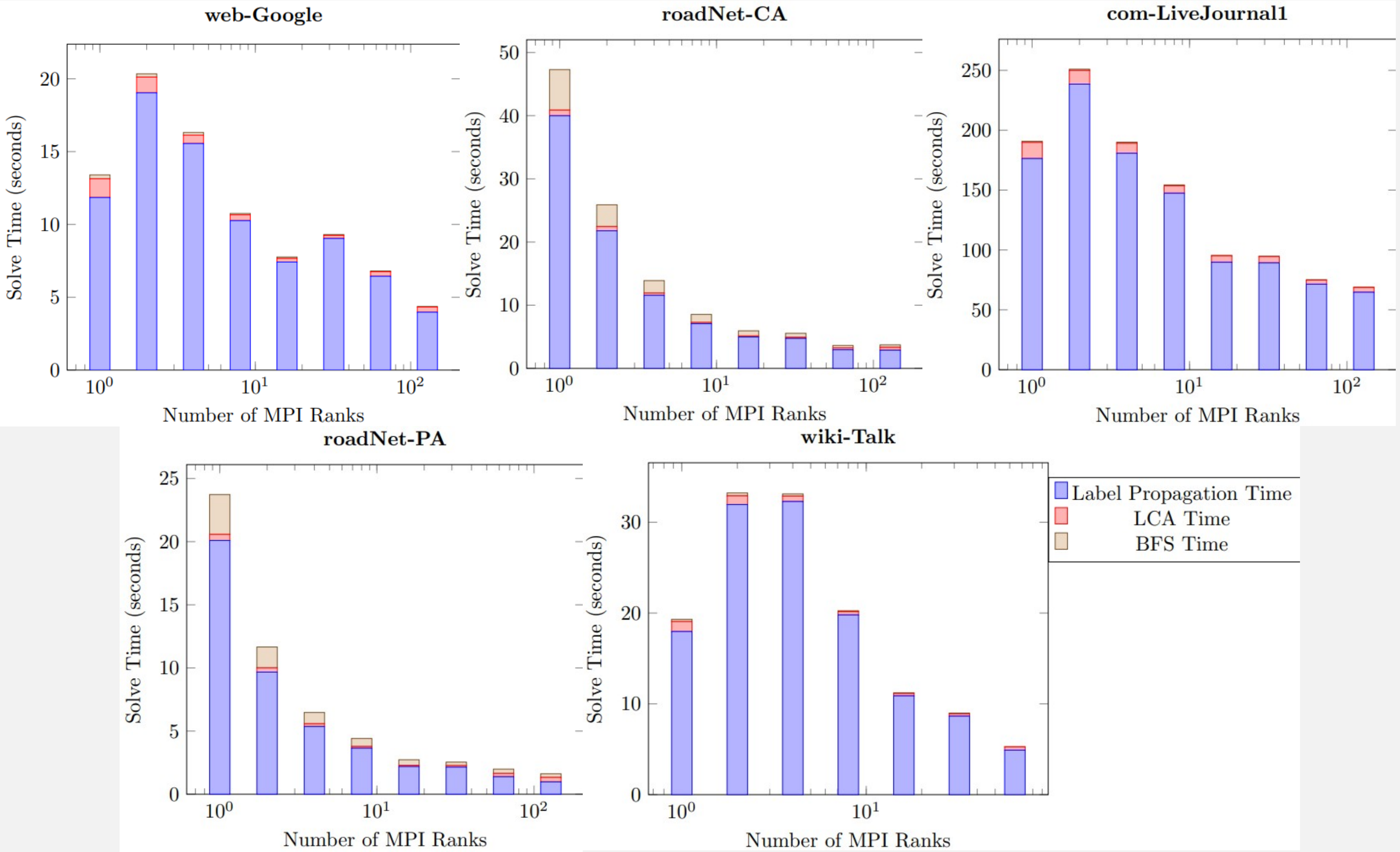
- Distributed label reduction is complex
  - Some IDs may belong to ghosts that do not have local data yet
  - Difficult to tell if a reduction can make progress or not before communicating
  - Reductions can require lookups across many different processes
- BCC-LR communication strategy
  - Reduce communication overhead by only communicating completely reduced labels
    - This involves nontrivial computation, and entails requesting information about remote vertices
- LCA traversal communication strategy
  - Communicate reduction traversals to the process that owns the vertices involved

Experimental Setup

- Experiments were run on RPI's DRP system
  - 64 nodes, with two eight-core 2.6 GHz Intel Xeon E5-2650 processors, 256 GB memory, connected with 56 Gb FDR Infiniband

Graph	**Type**	**# Verts**	**# Edges**	**Max Degree**	**BiCCs**
com-LiveJournal	Social	3.9M	69.3M	14.8K	594K
Wiki-Talk	Social	2.3M	5M	100K	34K
roadNet-CA	Road	1.9M	5.5M	12	327K
roadNet-PA	Road	1.0M	3M	9	194K
Web-Google	Web	0.9M	5.1M	6K	60K

Results


web-Google
roadNet-CA
com-LiveJournal1
roadNet-PA
wiki-Talk
Solve Time (seconds)
Number of MPI Ranks
Label Propagation Time
LCA Time
BFS Time

Future Work

- Explore alternative communication patterns for BCC-LR
- Use local parallelism for BCC-LR
- Compare against Biconnectivity algorithms such as Tarjan-Vishkin
- Explore more optimizations
- Contact Me: boglei@rpi.edu