Distributed Biconnectivity

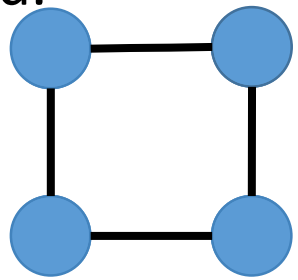
Ian Bogle[12] George Slota[1] Sivasankaran Rajamanickam[2] Karen Devine[2]

[1]Rensselaer Polytechnic Institute
Troy, NY

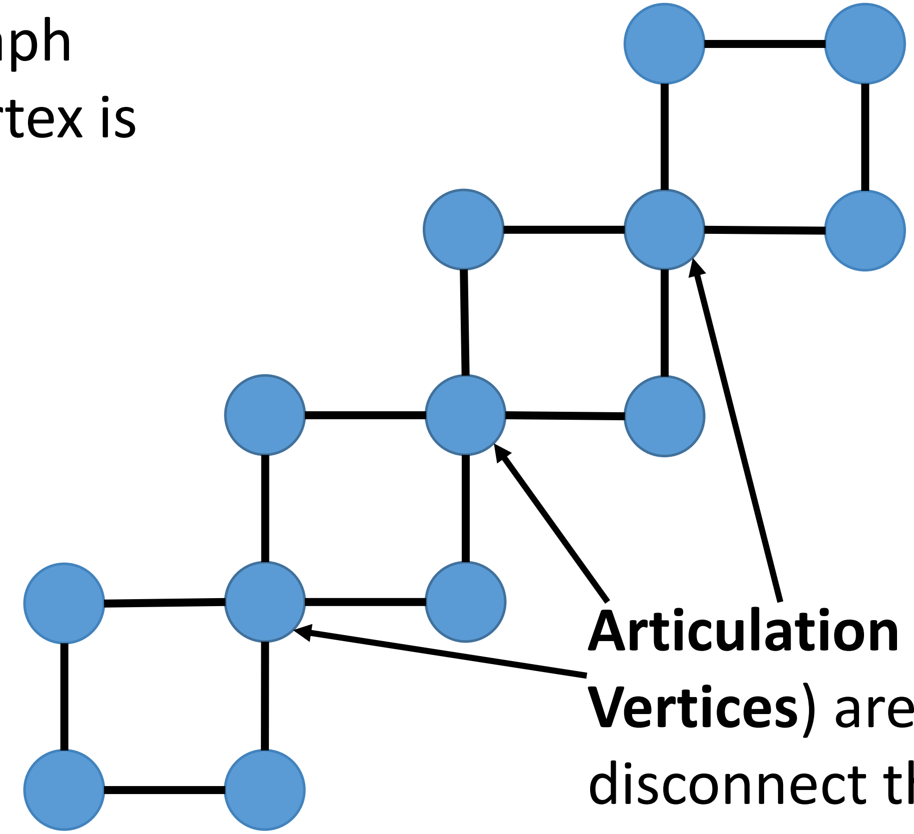
[2]Sandia National Laboratories
Albuquerque, NM

Graph Biconnectivity is a stronger version of graph connectivity

Biconnected Components of a graph remain connected if any single vertex is removed.



Graph Biconnectivity finds all **biconnected components** in an input graph



Articulation Points (or **Cut Vertices**) are vertices that disconnect the graph if removed.

Efficient Distributed Biconnectivity algorithms have practical applications

- Biconnectivity algorithms are useful for finding single points of failure in power and communications networks, as well as processing social networks
- Finding articulation points in meshes can help solvers converge
- We are not aware of any distributed parallel biconnectivity algorithms
- Efficient shared memory biconnectivity algorithms may not lend themselves to an efficient distributed memory implementation
- A new approach could lead to a more efficient distributed biconnectivity algorithm

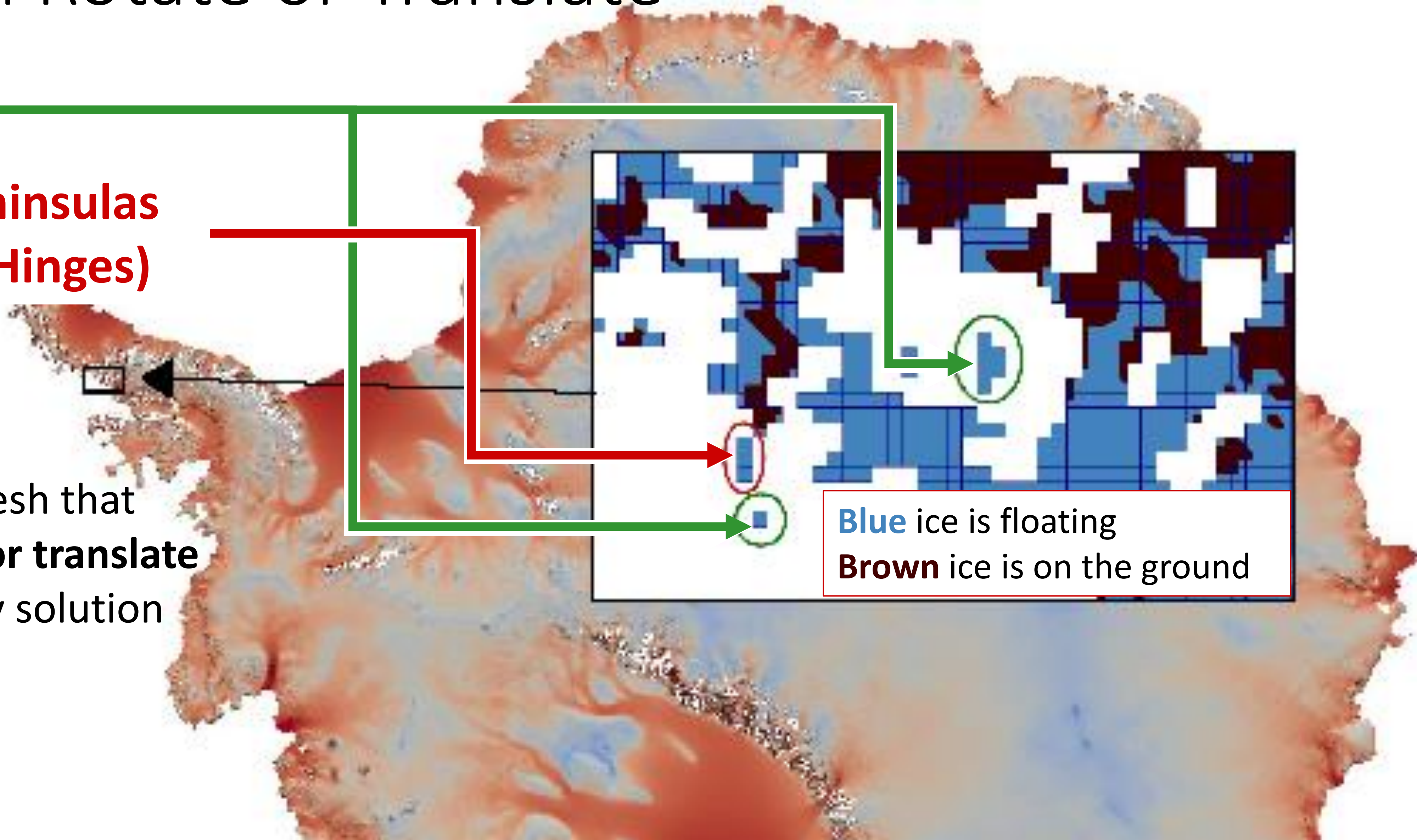
Our previous work implemented a distributed algorithm that solved a similar problem

- Determined whether certain parts of an Ice Sheet mesh were adequately connected
- During this work we realized we could use this Ice Sheet Connectivity algorithm (ICE-CONN) to find biconnected components in a general graph
- The distributed biconnectivity algorithm (BCC-ICE) we propose leverages the ICE-CONN algorithm

Degenerate Features are Parts of the Mesh That Can Rotate or Translate

Icebergs

Floating Peninsulas (Floating Hinges)



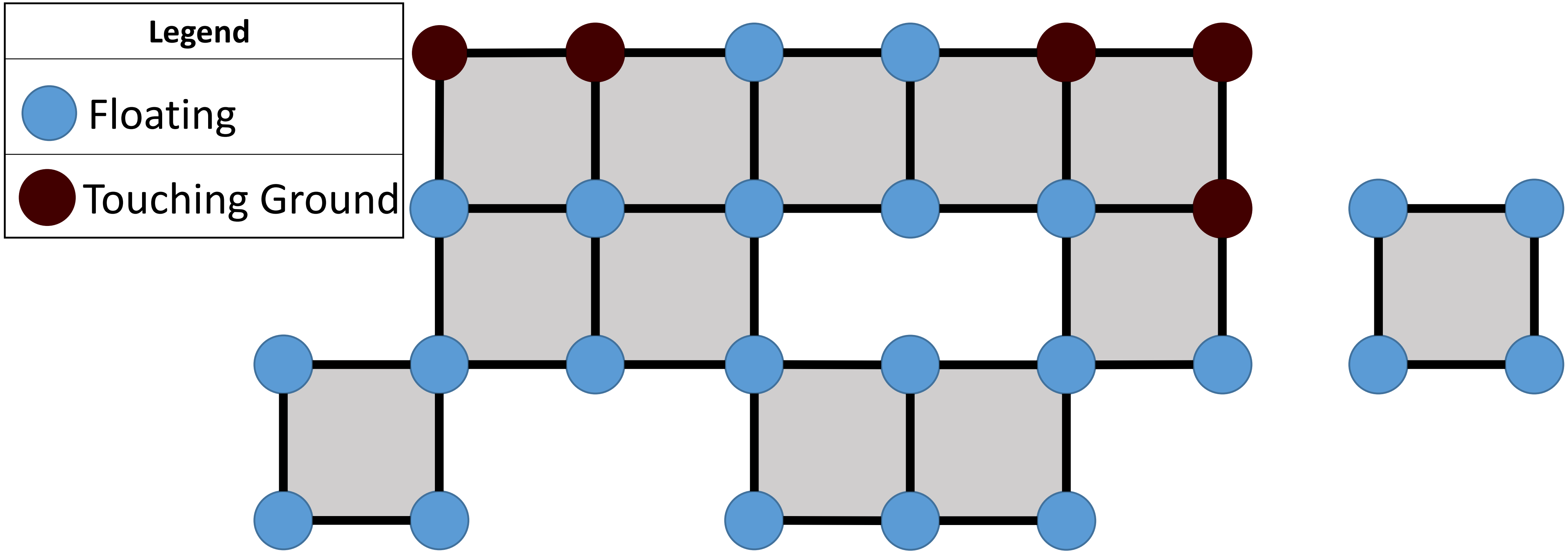
Any part of the mesh that can **freely rotate or translate** makes the velocity solution not unique

Previous Work: Efficiently Detect Degenerate Mesh Features

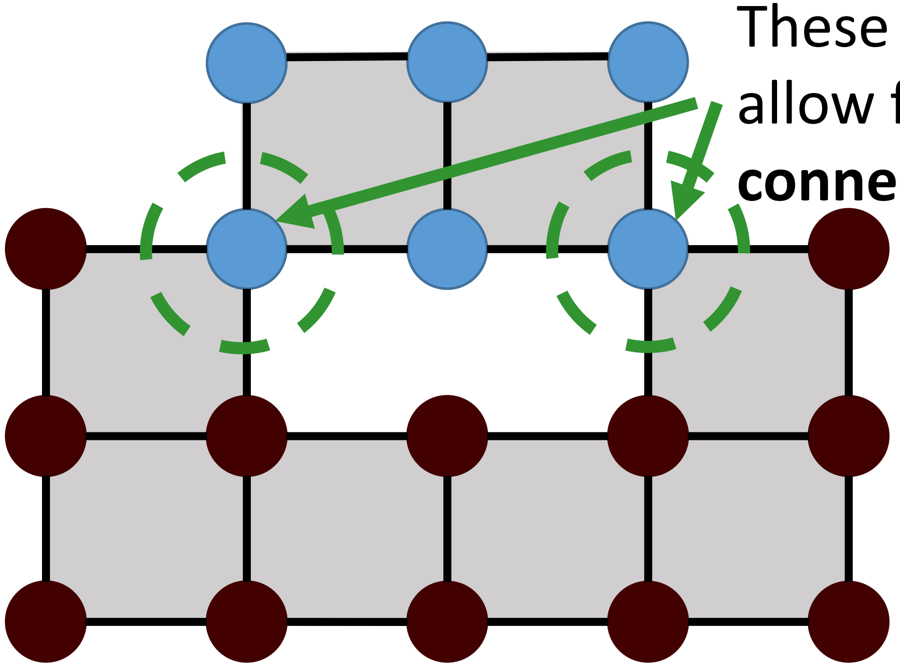
- Ice sheet simulations fail to converge due to features like hinged peninsulas and icebergs in meshes
- New algorithm that detects all degenerate features
- Distributed memory implementation provides good strong scaling and weak scaling up to 4096 processors
- Detection takes at most 0.4% of a simulation step's runtime
- 46,000x faster than previously used preprocessing on highest resolution meshes

This work won a Best Paper award, published in Bogle, Devine, Perego, Rajamanickam, and Slota. "A Parallel Graph Algorithm for Detecting Mesh Singularities in Distributed Memory Ice Sheet Simulations." *Proceedings of the 48th International Conference on Parallel Processing.* 2019.

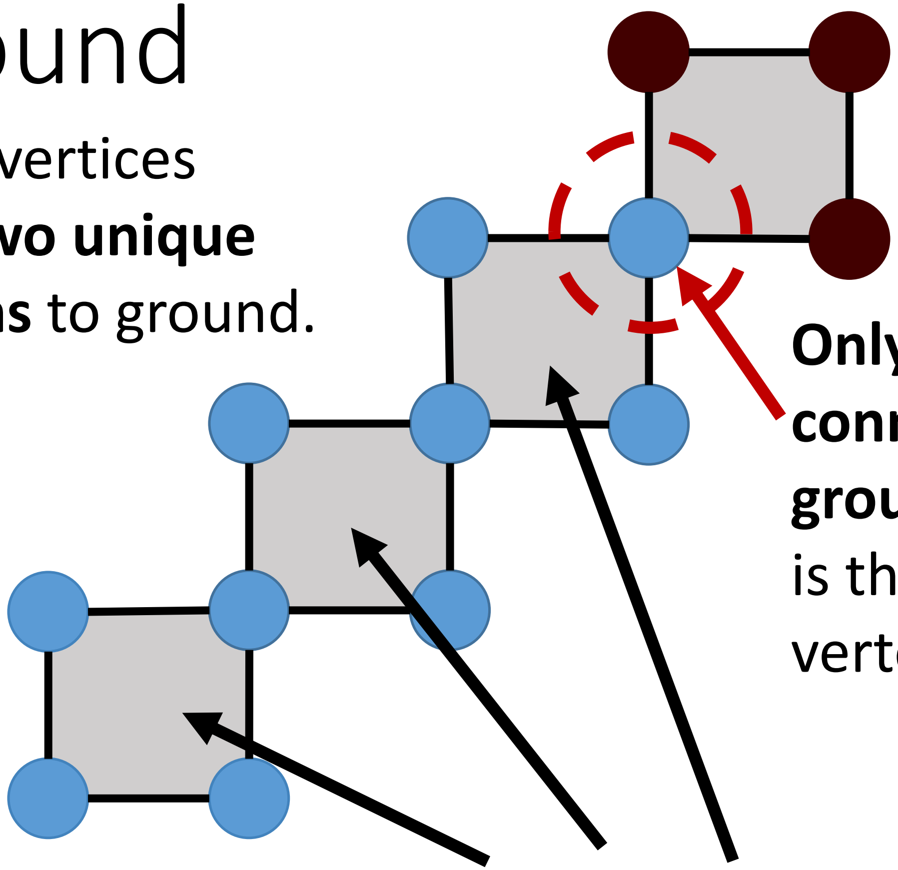
Application Provides a Mesh and Grounding Information



Degenerate Mesh Features Have at Most One Connection to the Ground



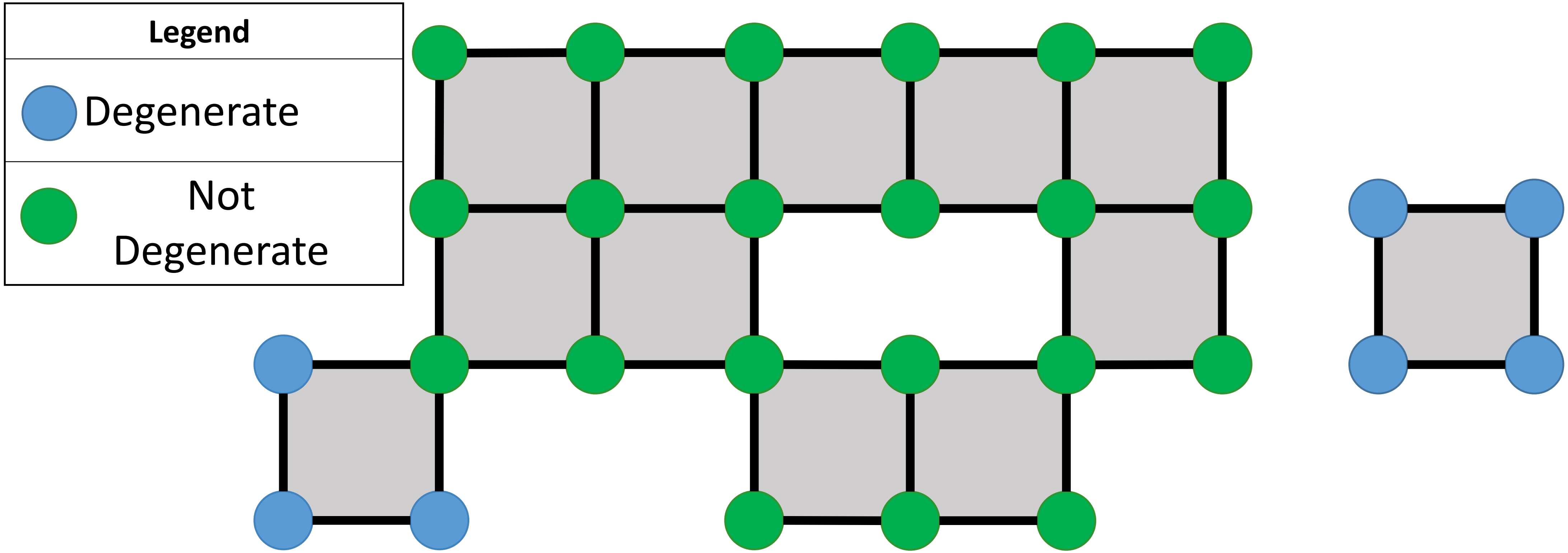
There are **no** Degenerate Features



Vertex that is floating in the water

Vertex that is touching the ground

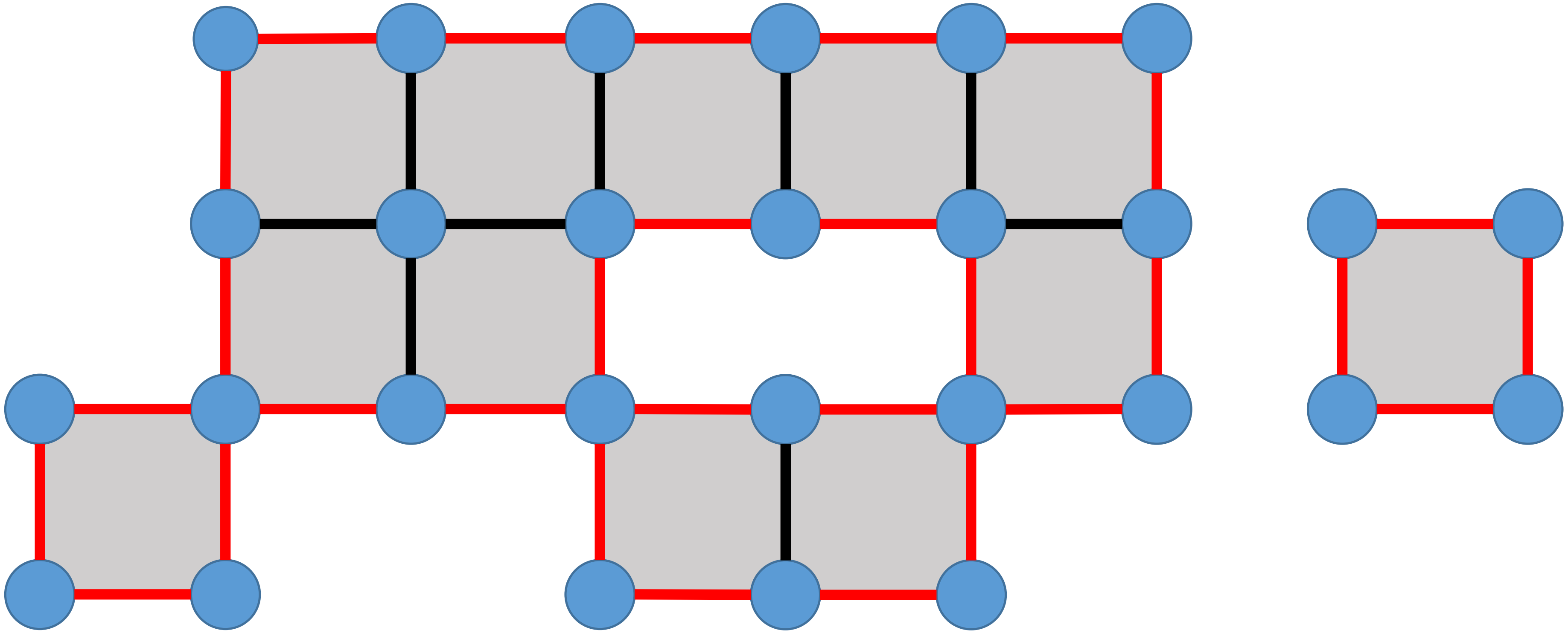
We Identify Parts of the Mesh with No Degenerate Features



The ICE-CONN Algorithm Propagates Grounding Information Through the Mesh

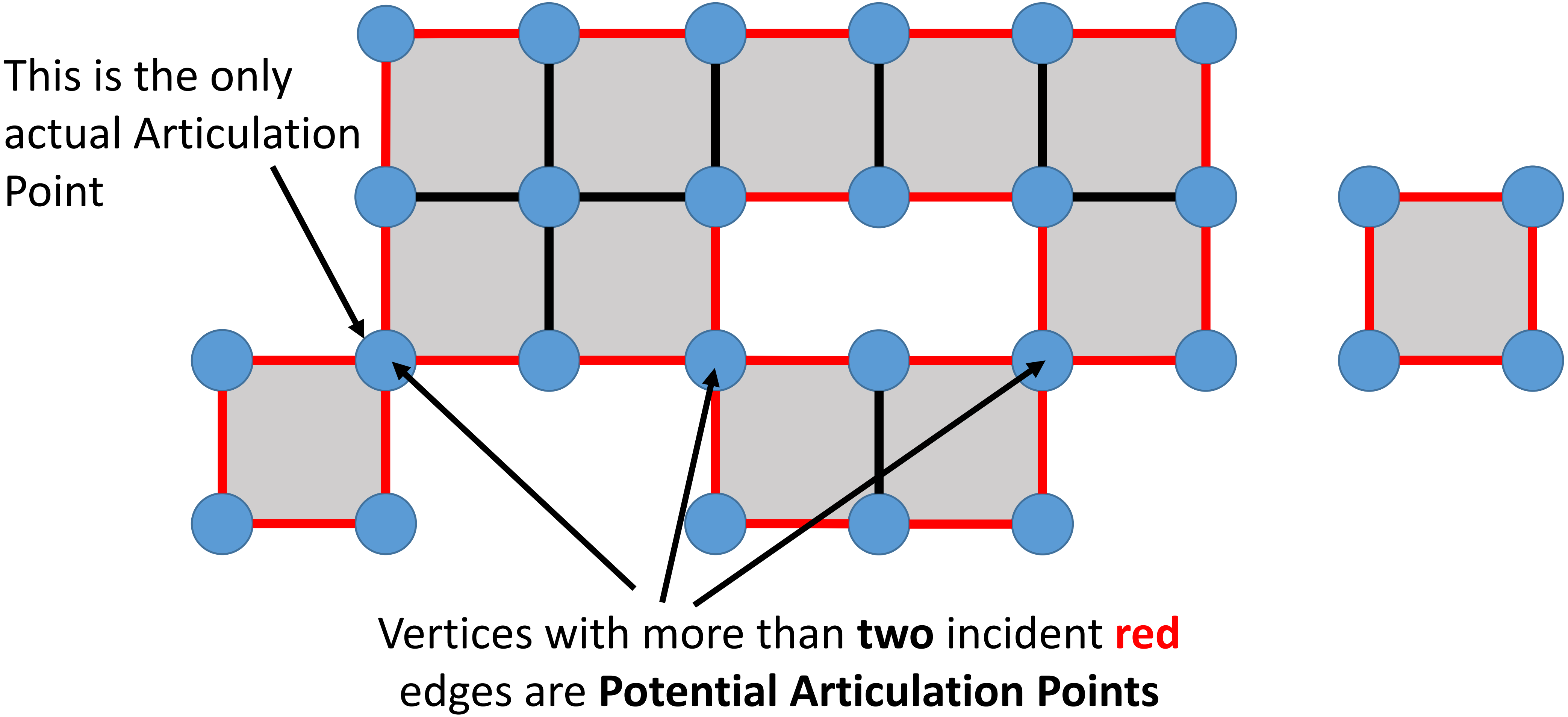
- The ICE-CONN algorithm has two steps:
  - Find Potential Articulation Points
  - Propagate Grounding Information
- We exploit mesh boundary information to identify potential articulation points
- Propagating grounding information reveals degenerate mesh features
- **Note**: Examples show quad meshes, but the approach works with triangular meshes as well.

Step 1: Find Potential Articulation Points



Application identifies **boundary edges** at interfaces between ice and water

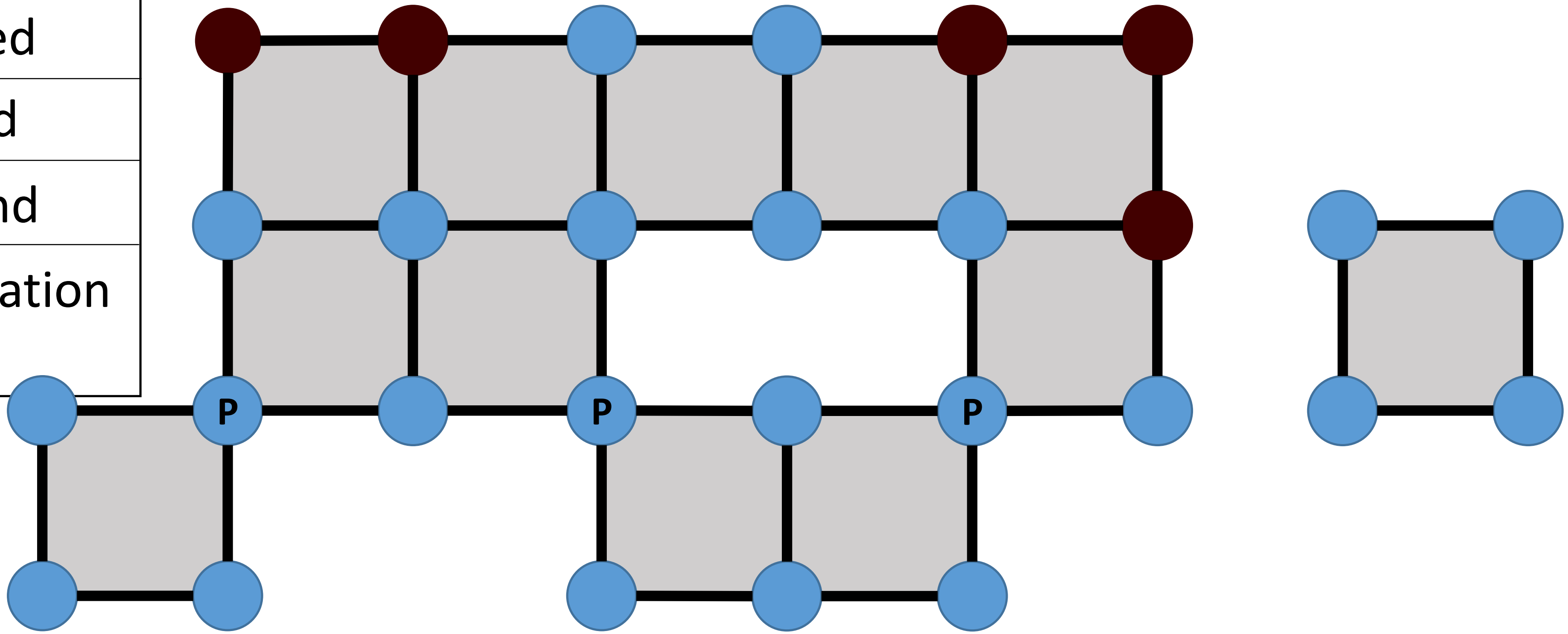
Step 1: Find Potential Articulation Points



Step 2: Propagate Grounding Information

Legend
Floating
Initially Grounded
1 Path to Ground
2 Paths to Ground
P Potential Articulation Point

Start with grounding given from the application

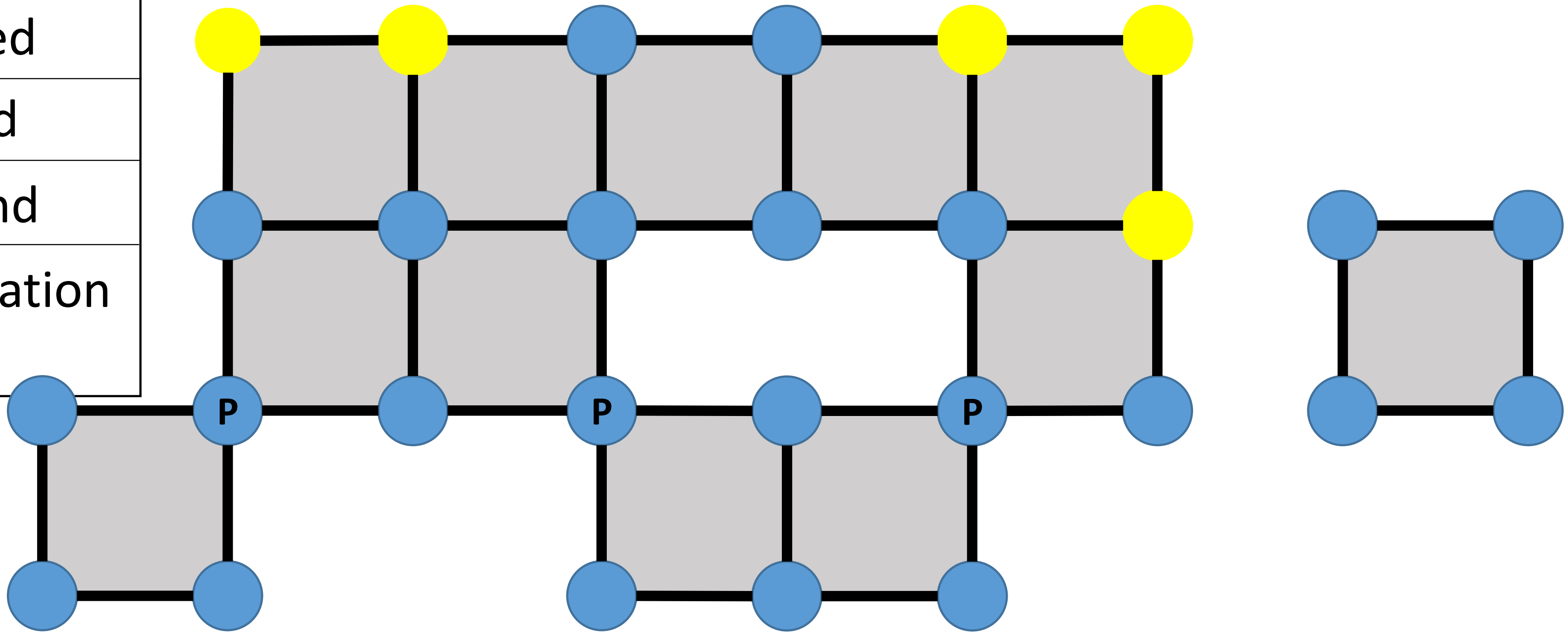


Step 2: Propagate Grounding Information

Legend

- Floating
- Initially Grounded
- 1 Path to Ground
- 2 Paths to Ground
- P Potential Articulation Point

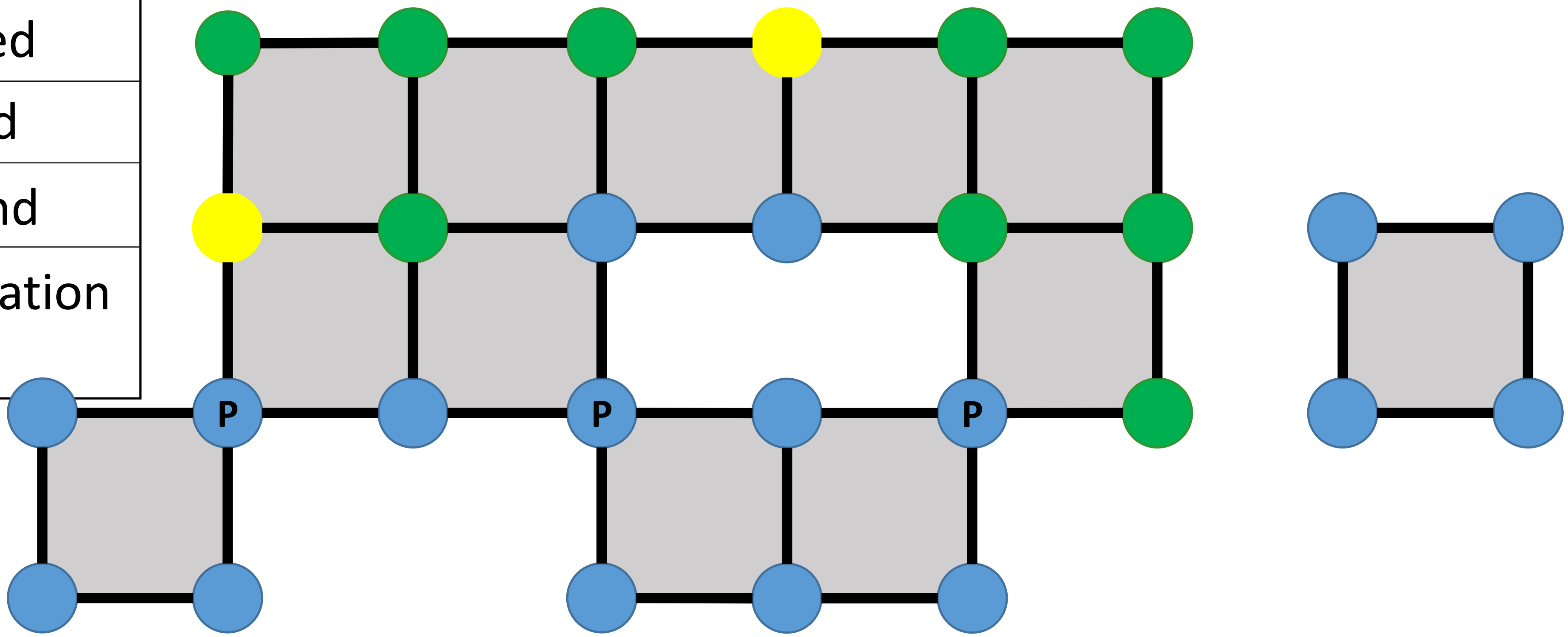
Initially grounded vertices have one path to ground



Step 2: Propagate Grounding Information

Legend
Floating
Initially Grounded
1 Path to Ground
2 Paths to Ground
P Potential Articulation Point

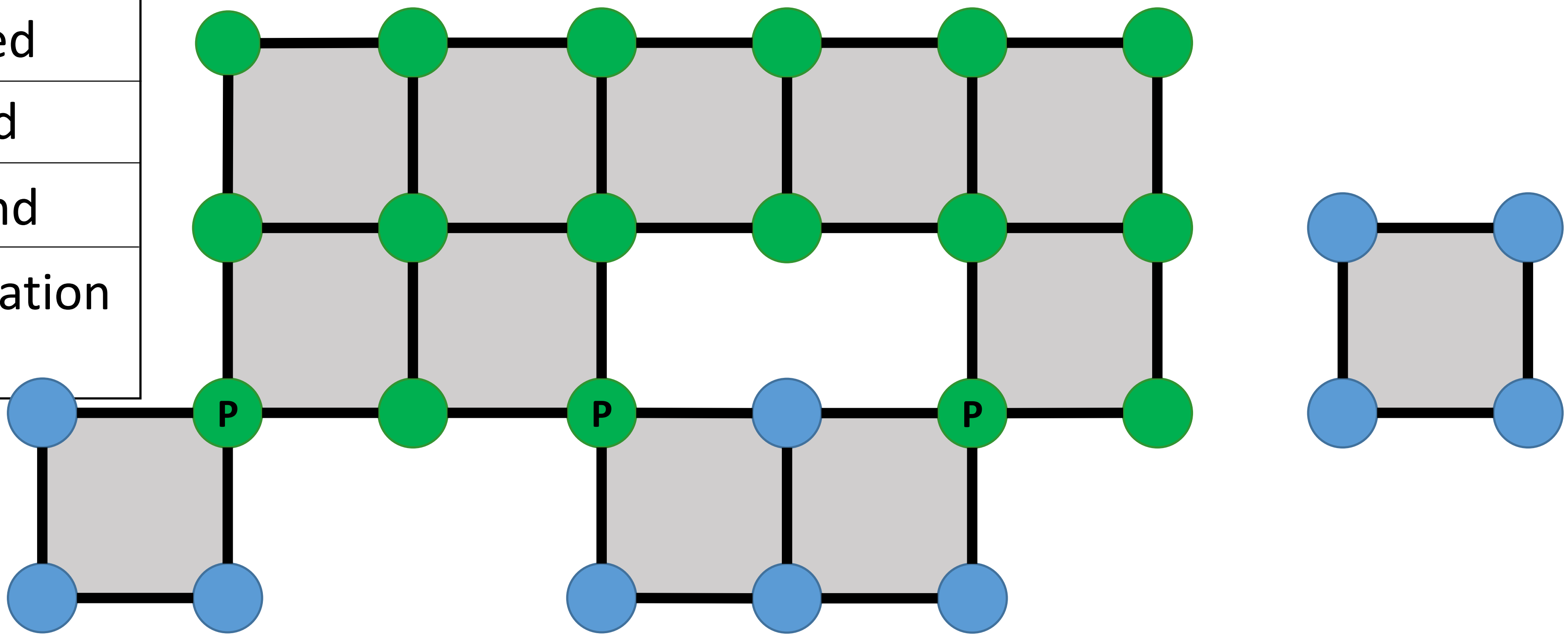
Propagate only from vertices that have changed color



Step 2: Propagate Grounding Information

Legend
Floating
Initially Grounded
1 Path to Ground
2 Paths to Ground
P Potential Articulation Point

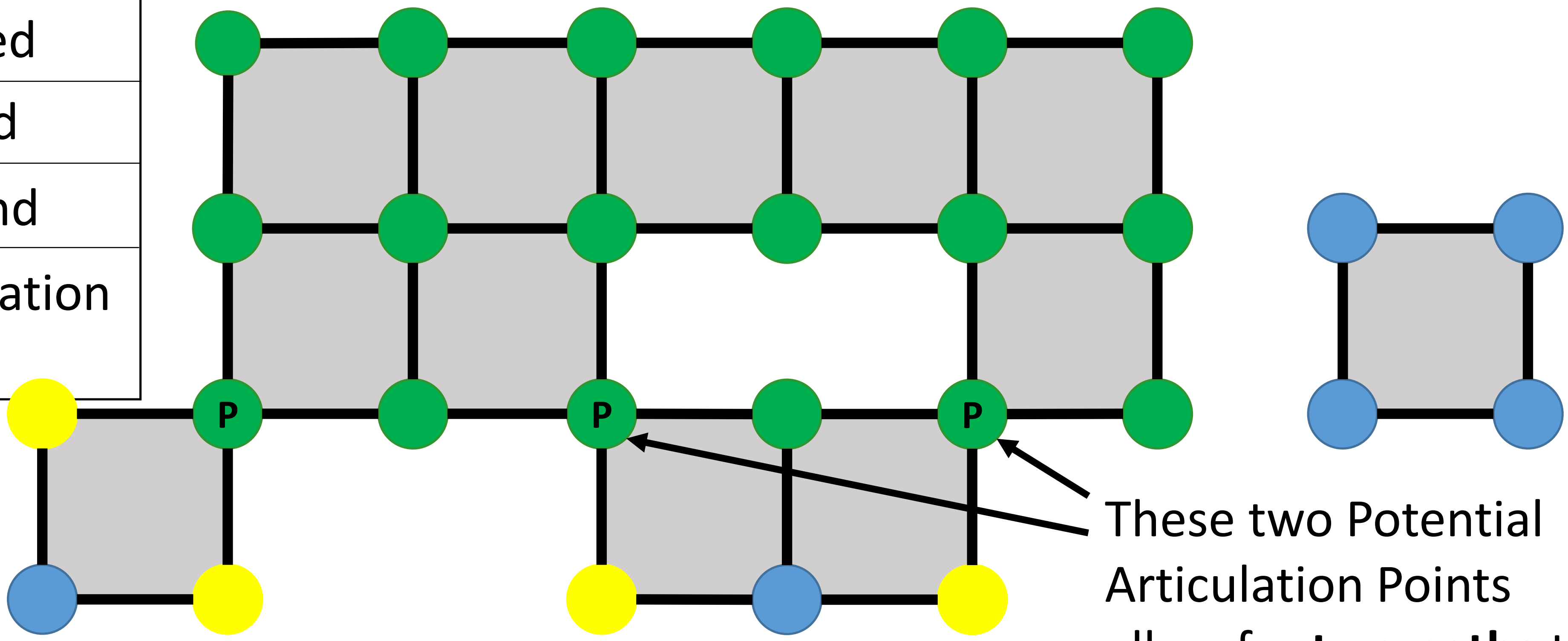
Stop the propagation at the Potential Articulation Points



Step 2: Propagate Grounding Information

Legend

- Floating
- Initially Grounded
- 1 Path to Ground
- 2 Paths to Ground
- P Potential Articulation Point

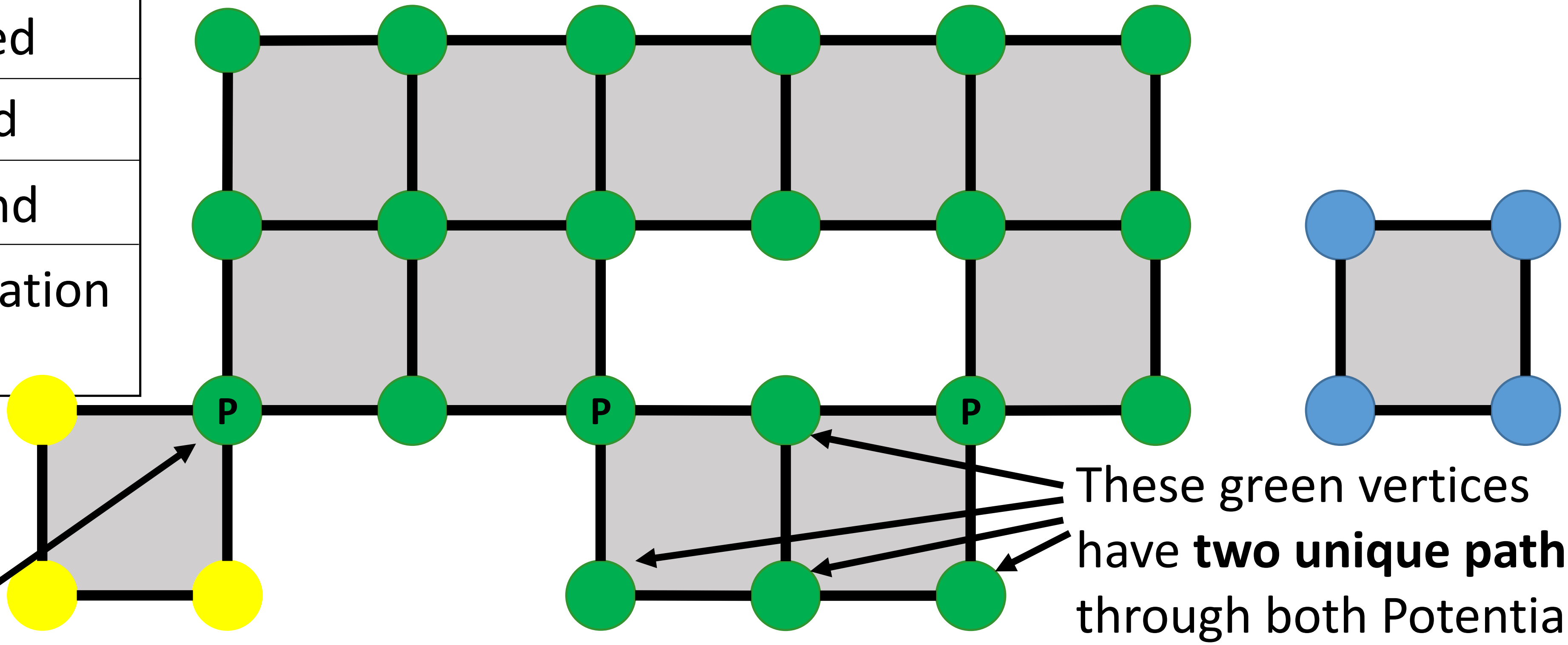


These two Potential Articulation Points allow for **two paths** to ground

Final Result of the ICE-CONN Algorithm

Legend
Floating
Initially Grounded
1 Path to Ground
2 Paths to Ground
P Potential Articulation Point

Keep all vertices with **two unique paths** to the ground



The yellow vertices only have **one unique path** through this vertex

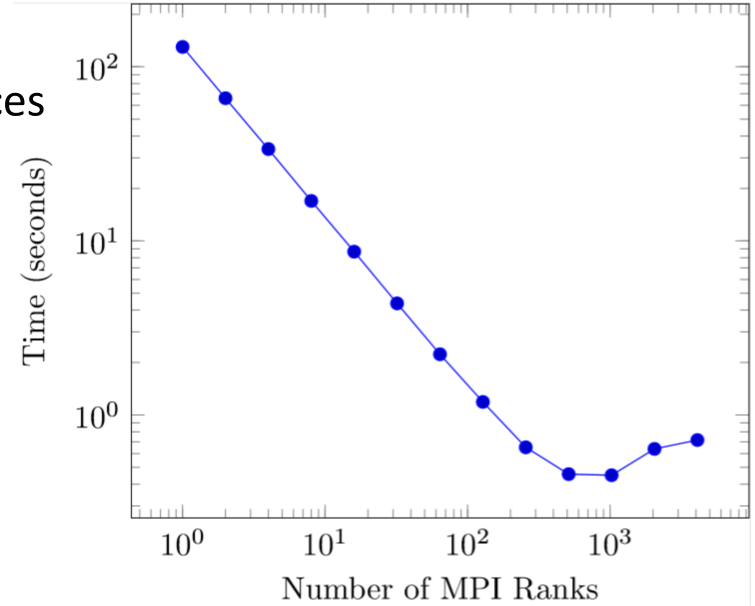
These green vertices have **two unique paths** through both Potential Articulation Points

The ICE-CONN Algorithm scales well for mesh inputs

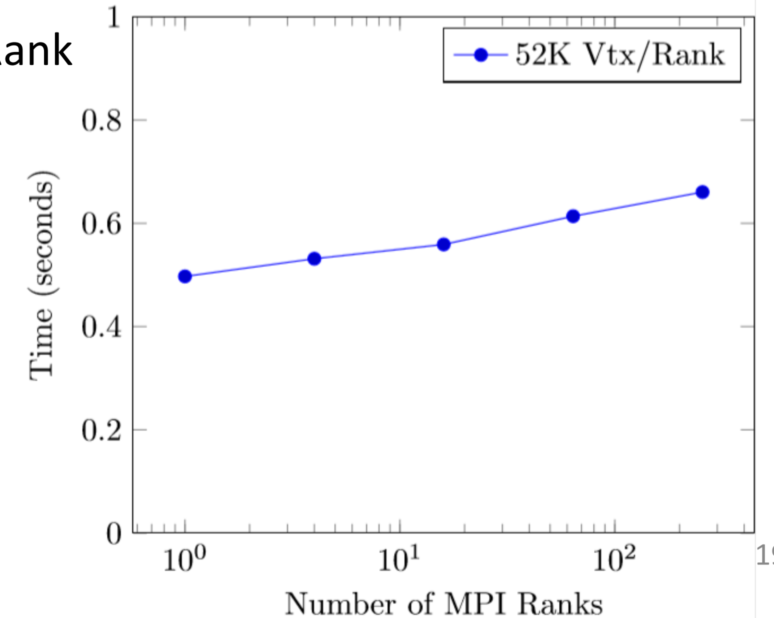
# Vertices	Our Algorithm (Ranks)	Matlab Preprocessing
52,465	0.0176 s (6)	1.04 s
210,170	0.0217 s (24)	5.65 s
841,346	0.0414 s (96)	34.60 s
3,368,275	0.0407 s (384)	245.00 s
13,479,076	0.0561 s (1536)	2630.00 s

Comparison against Matlab preprocessing in Tuminaro, Perego, Tezaur, Salinger, Price. “A matrix dependent/algebraic multigrid approach for extruded meshes with applications to ice sheet modeling”. *SIAM Journal on Scientific Computing* 38.5 (2016): C504-C532

Strong scaling
13 Million vertices



Weak scaling
52k Vertices/Rank



With slight modifications, the ICE-CONN algorithm generalizes to solve graph biconnectivity

- We ground two neighboring vertices – the smallest biconnected component
- We use a more general heuristic to find potential articulation points
- We use the ICE-CONN to find biconnected components iteratively
- This distributed biconnectivity algorithm will be referred to as BCC-ICE

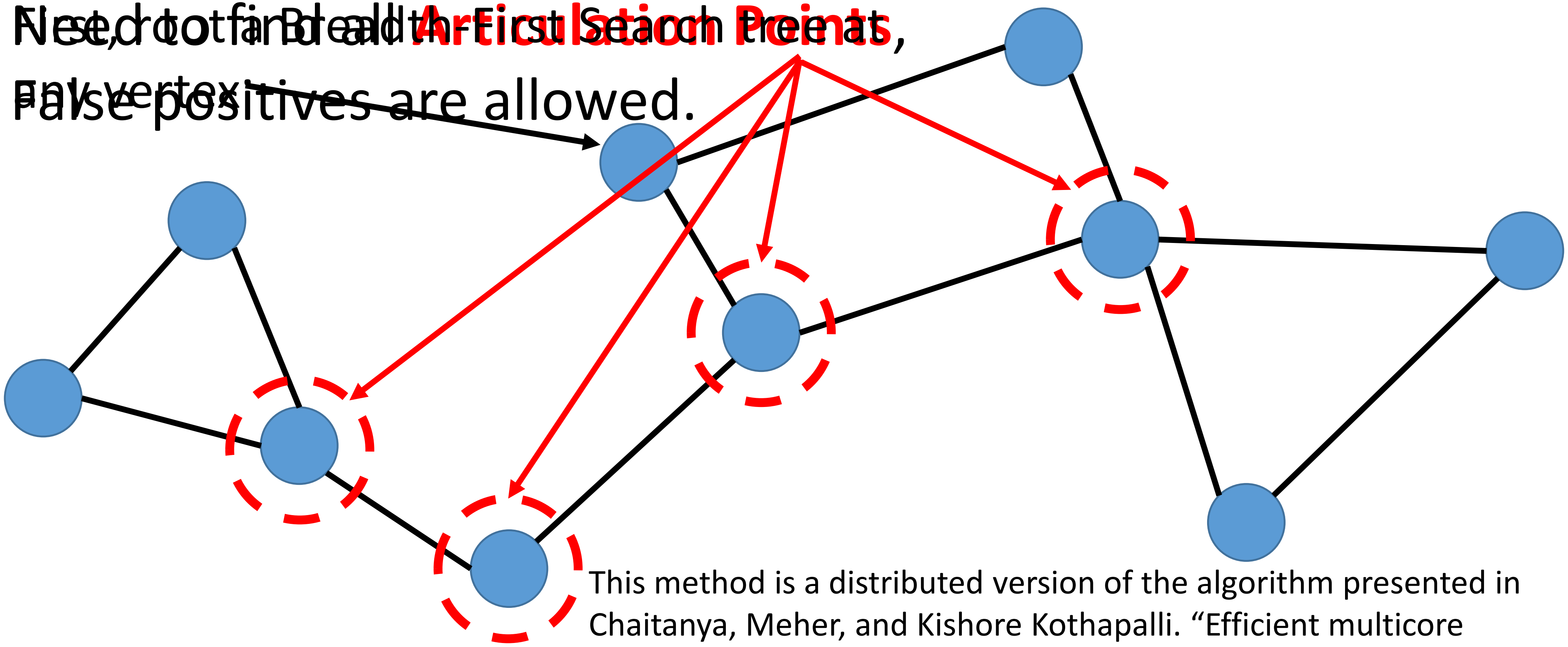
We use a novel distributed LCA algorithm to find potential articulation points

Need to find all **Articulation Points**,

False positives are allowed.

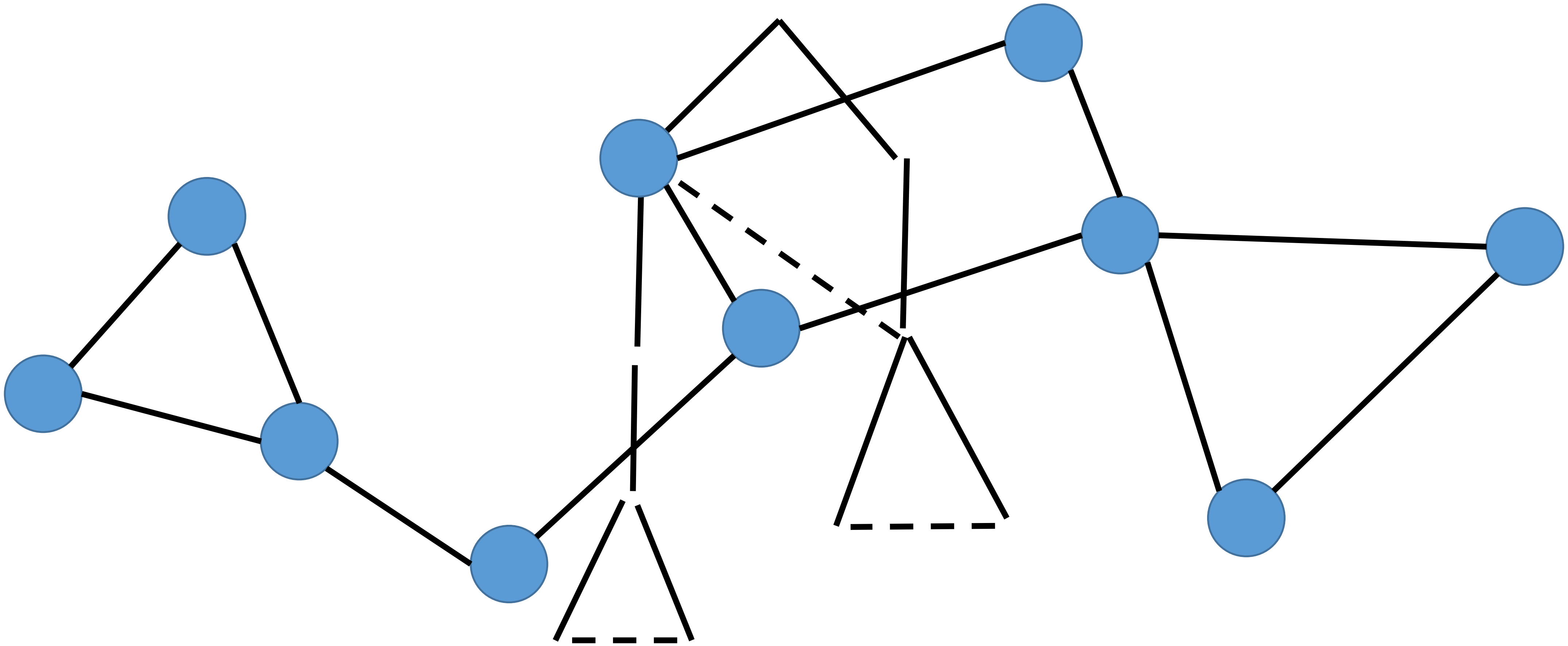
First, root a Breadth-First Search tree at

any vertex

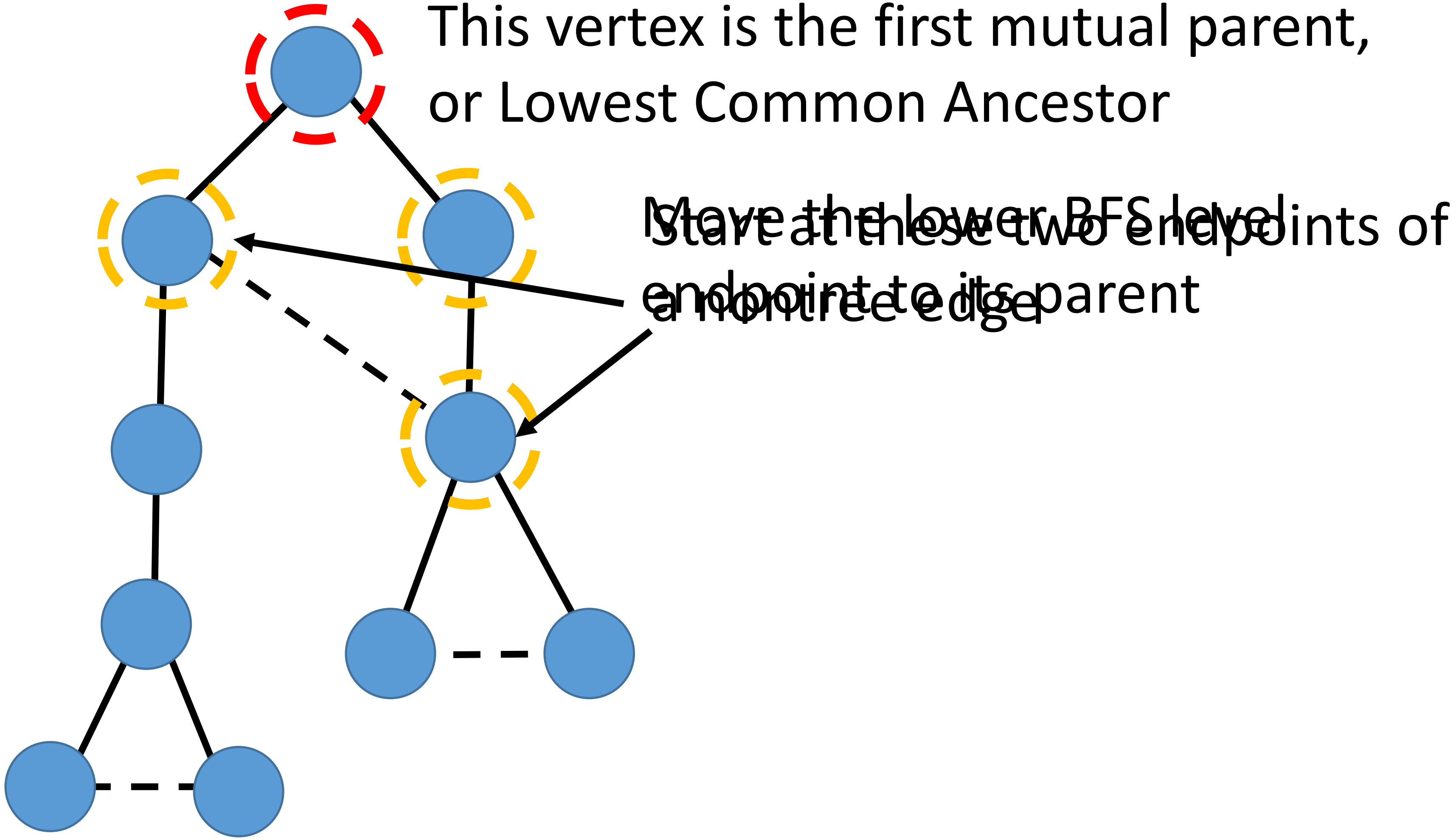


This method is a distributed version of the algorithm presented in Chaitanya, Meher, and Kishore Kothapalli. "Efficient multicore algorithms for identifying biconnected components." *International Journal of Networking and Computing* 6.1 (2016): 87-106

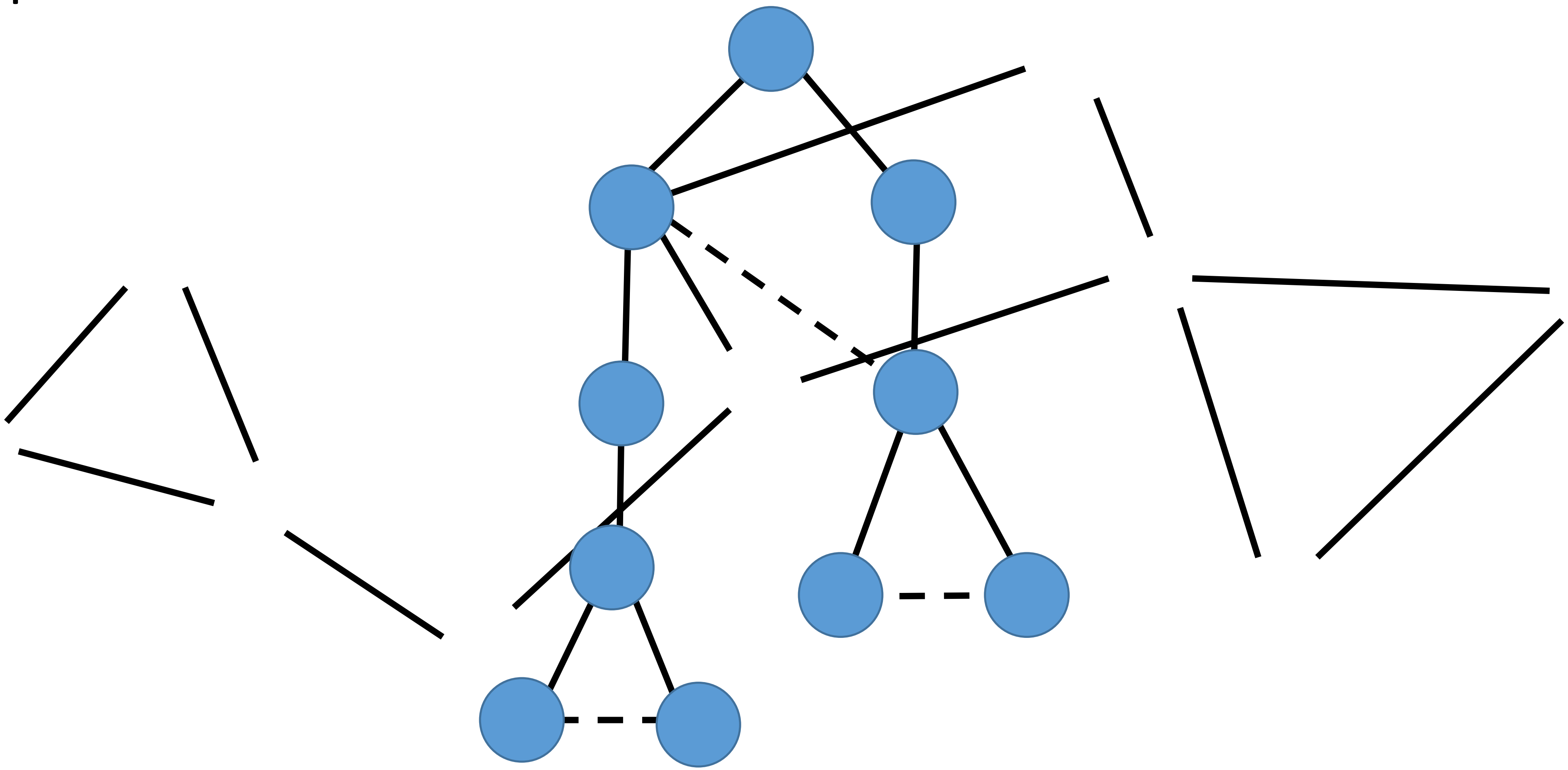
Lowest Common Ancestor(LCA) traversals start from non-tree edges and end at the first mutual parent



Lowest Common Ancestor(LCA) traversals start from non-tree edges and end at the first mutual parent



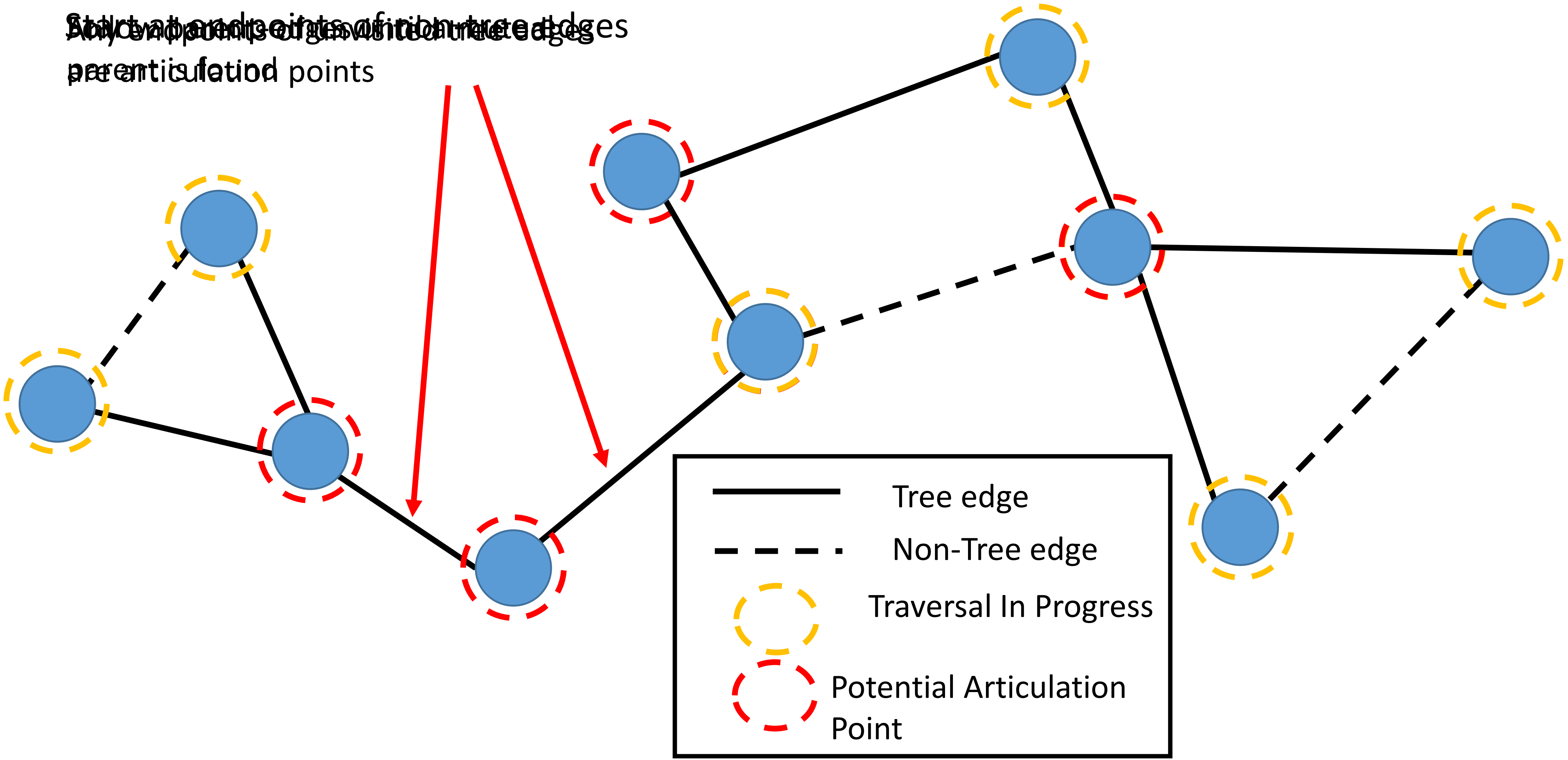
Lowest Common Ancestor(LCA) traversals start from non-tree edges and end at the first mutual parent



We use a novel distributed LCA algorithm to find potential articulation points

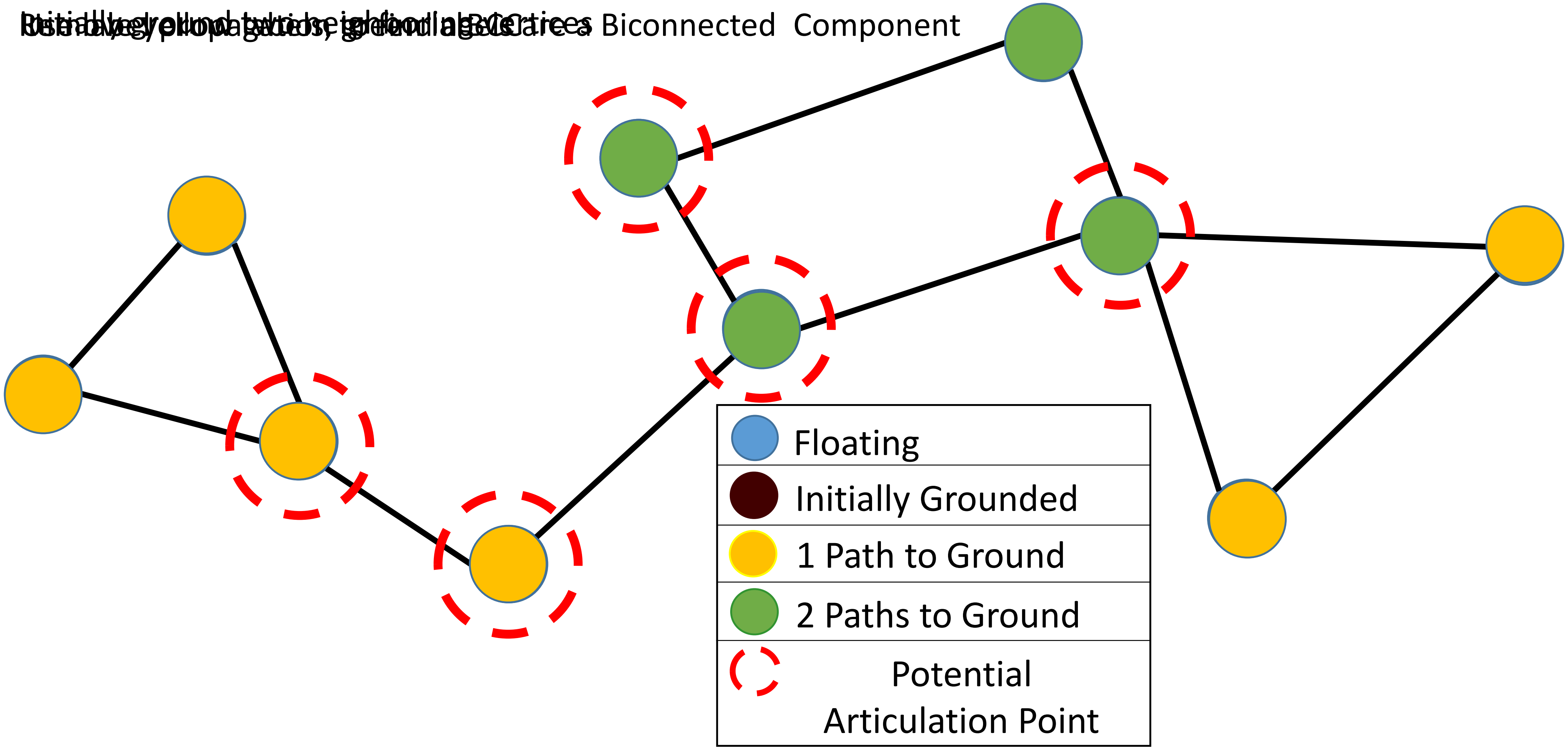
Start at endpoints of non-tree edges

Any endpoints of unvisited tree edges are articulation points



We use the ICE-CONN algorithm to find Biconnected Components Iteratively

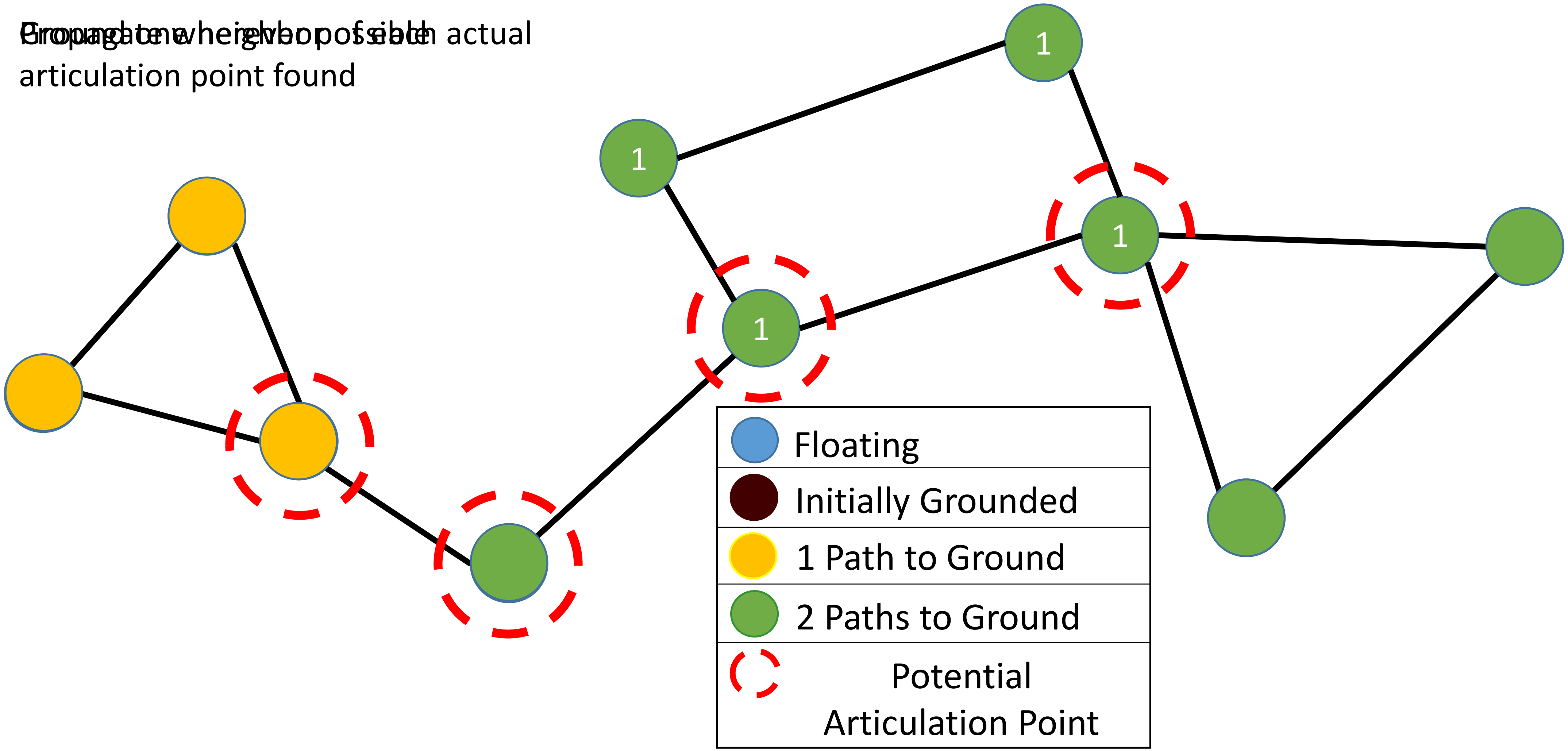
a Biconnected Component



We use the ICE-CONN algorithm to find Biconnected Components Iteratively

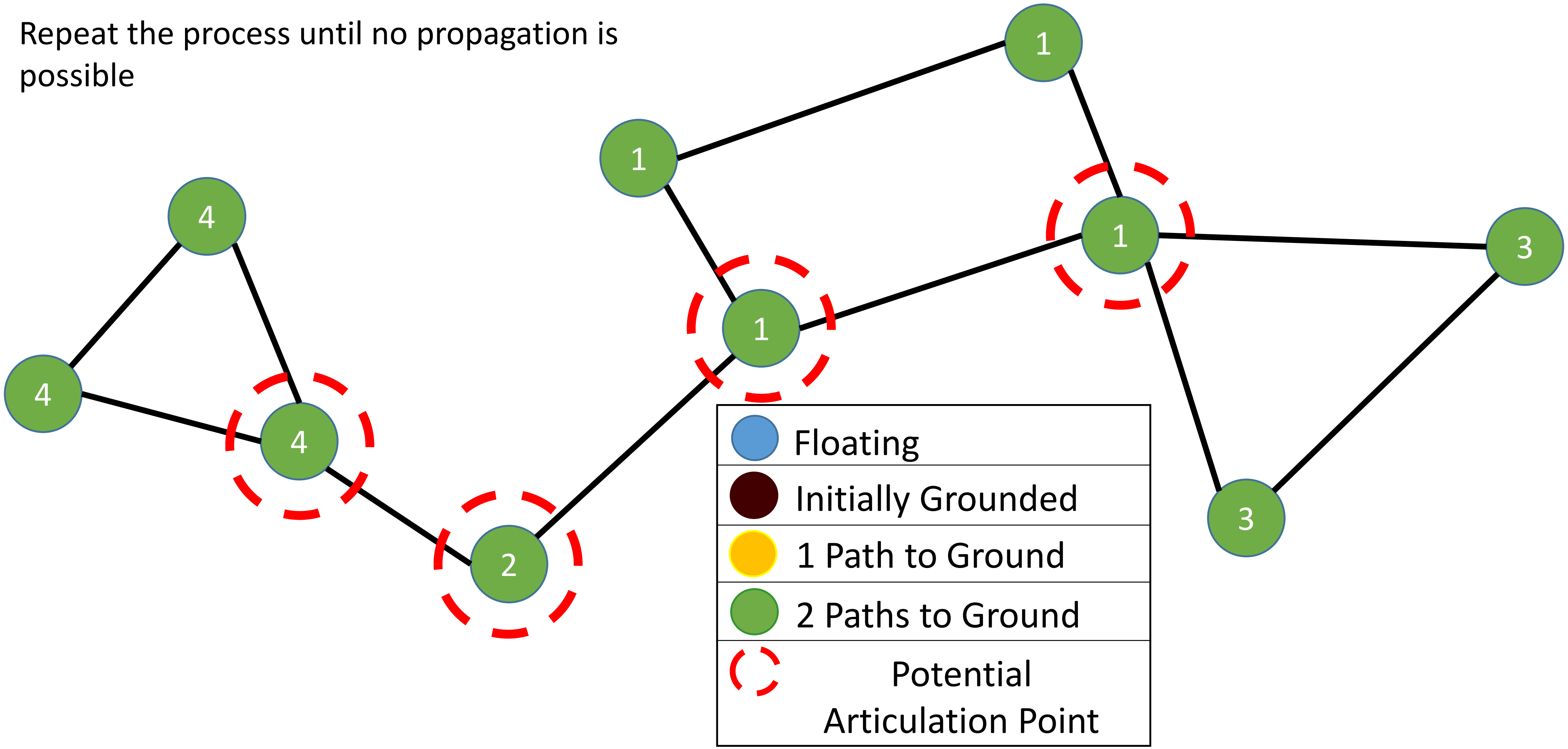
Propagate where possible

Ground neighbor of each actual articulation point found



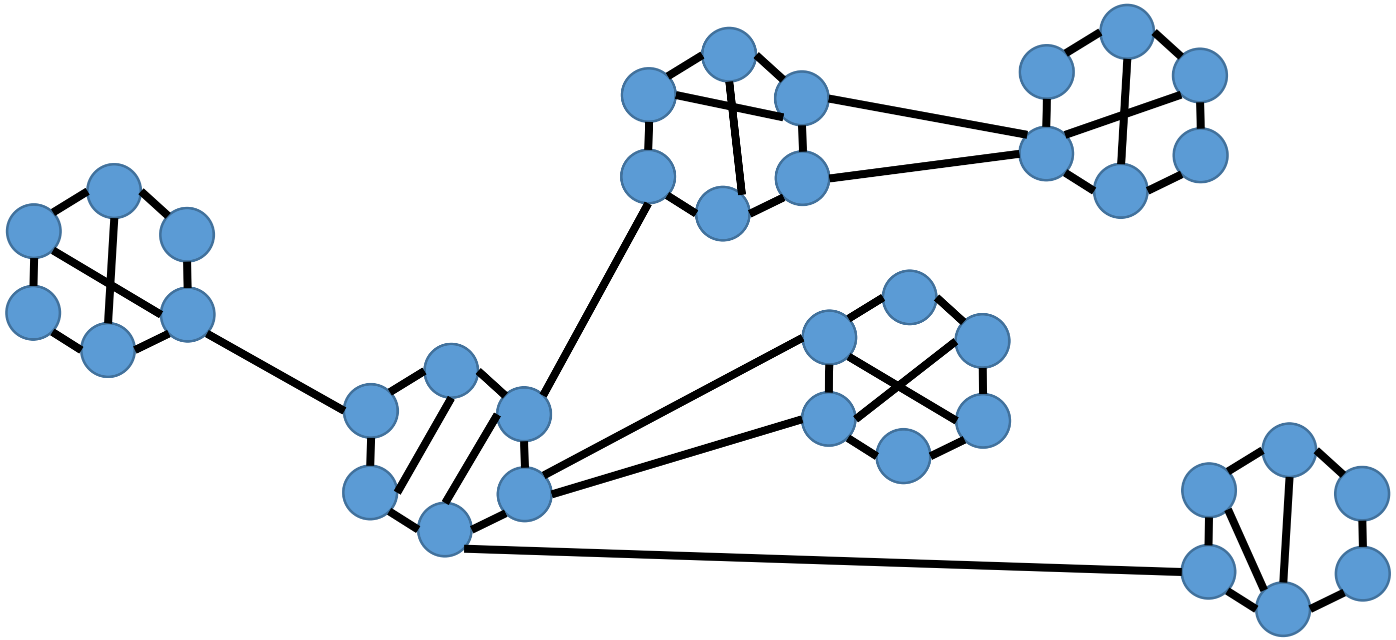
We use the ICE-CONN algorithm to find Biconnected Components Iteratively

Repeat the process until no propagation is possible



Experimental Setup

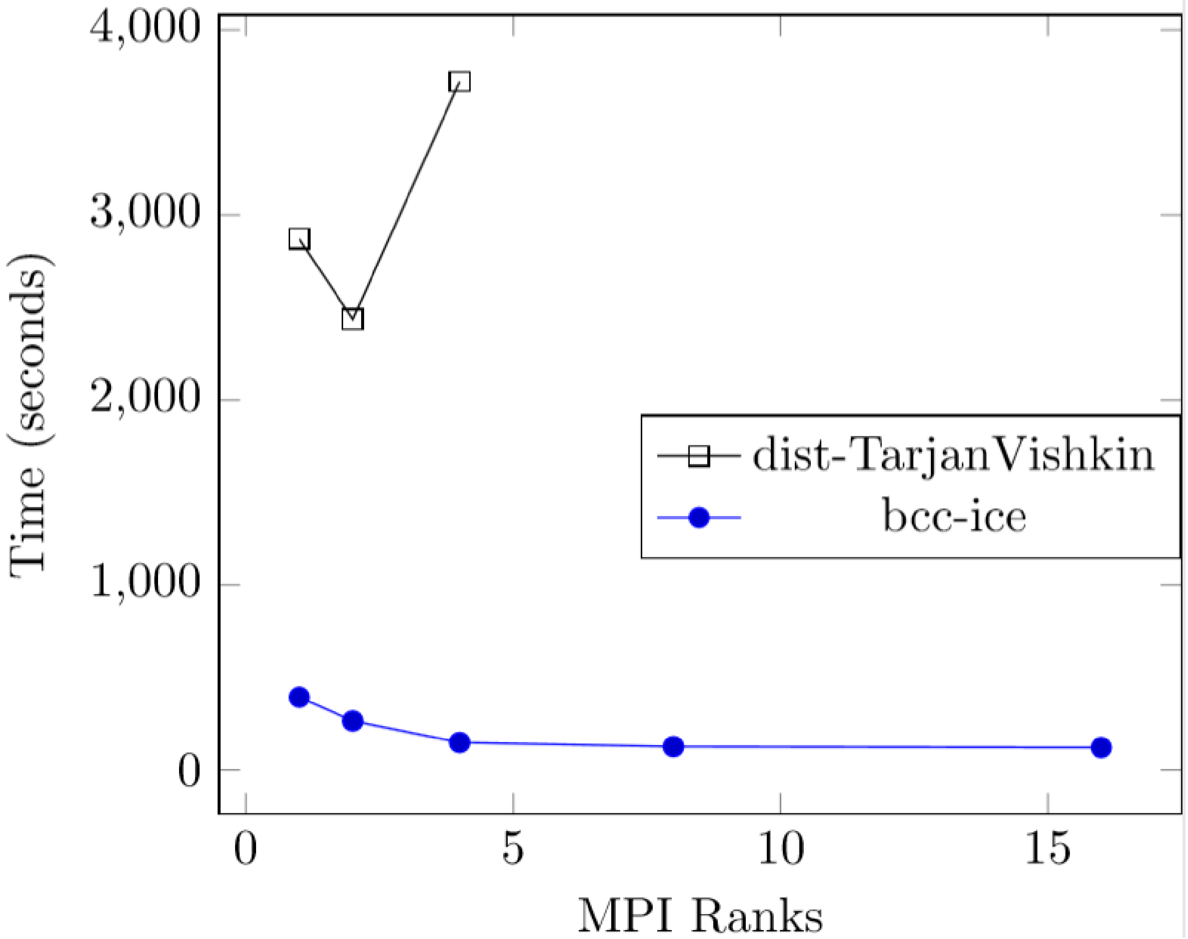
- Tests were run on Sandia National Labs' Blake platform
  - 40 nodes equipped with dual socket Intel Xeon Platinum CPUs.
- We generated synthetic graphs with a known number of biconnected components to validate our implementations.



We implemented the shared-memory Tarjan-Vishkin algorithm in distributed memory as a baseline

- Presented by Tarjan and Vishkin, 1985
  - Tarjan, Robert E., and Uzi Vishkin. “An efficient parallel biconnectivity algorithm.” *SIAM Journal on Computing* 14.4 (1985): 862-874.
- Optimal in a shared-memory architecture
- Requires a Breadth-First-Search, the computation of preorder labels and the number of descendants for each vertex
- Constructs an auxiliary graph
  - # Vtx in auxiliary graph = # edges in original graph
  - Filter edges based on values computed for each vertex
- Connected components in auxiliary graph correspond to biconnected components in the original graph

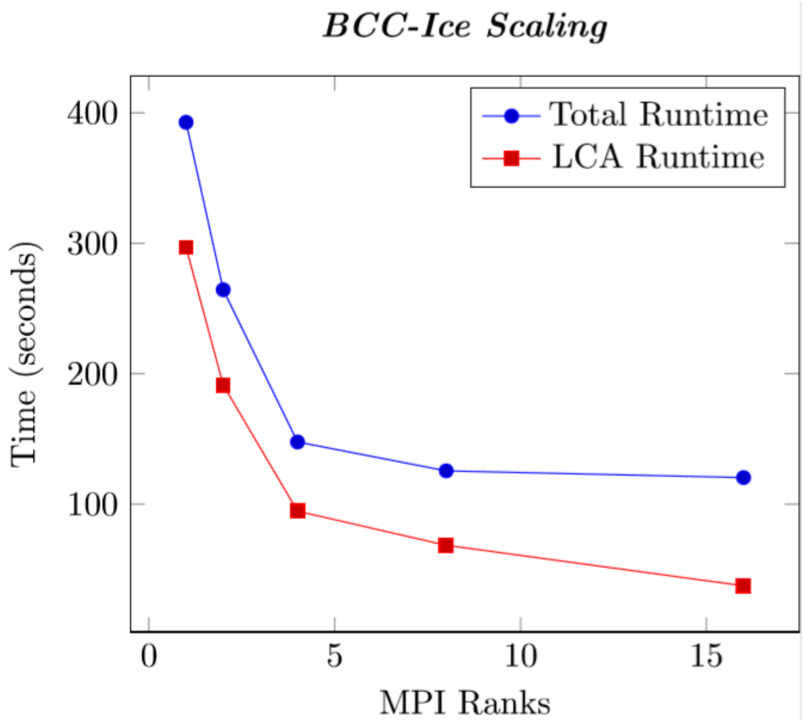
Our BCC-ICE approach outperforms our distributed implementation of the Tarjan-Vishkin algorithm



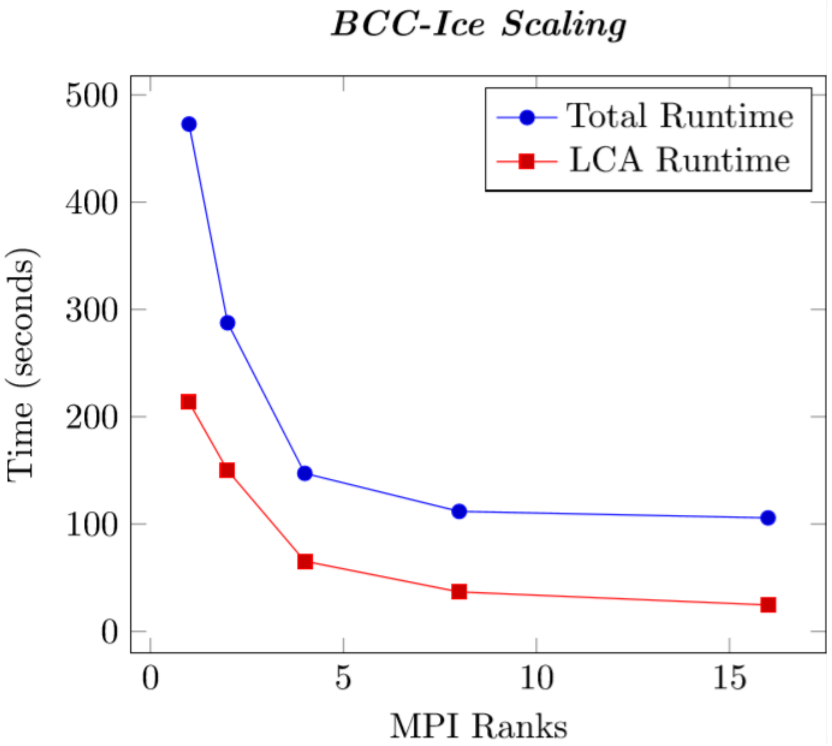
- 10 Million Vertices
  - Avg degree 16
  - 10 Biconnected Components
- Our Tarjan-Vishkin implementation does not scale well
  - Constructing the auxiliary graph is expensive in distributed memory
  - Final labeling of the input graph requires communication and is nontrivial

Our BCC-ICE algorithm's scaling depends on the structure of the input graph

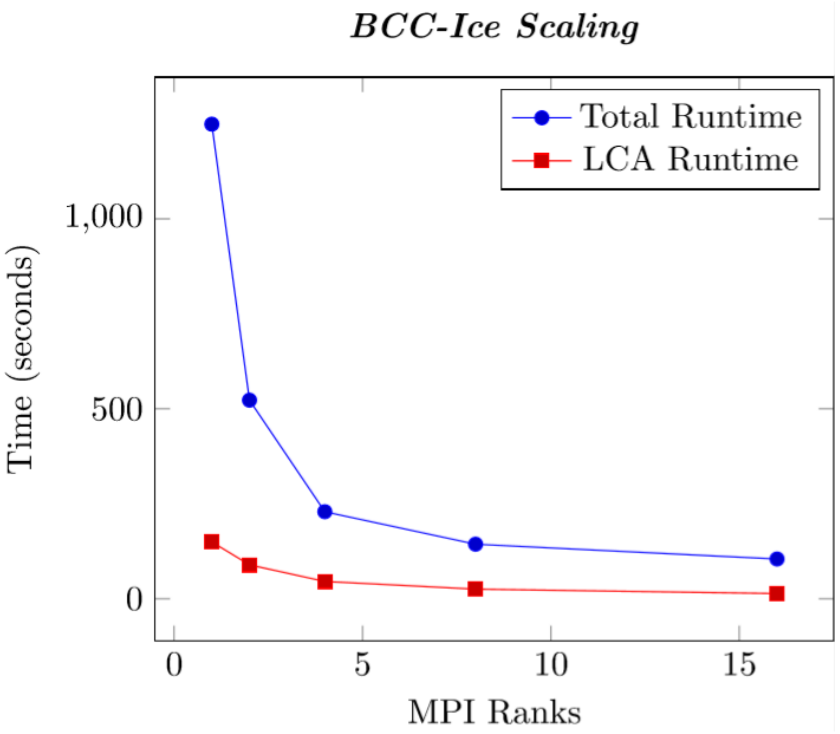
10 Biconnected Components



100 Biconnected Components



1000 Biconnected Components



All inputs have 10 Million vertices and average degree 16

Conclusions and Future Work

- Our ICE-CONN algorithm efficiently detects degenerate features of ice-sheet meshes in distributed memory.
- We generalize ICE-CONN to solve biconnectivity in distributed memory
- This direct generalization (BCC-ICE) is more efficient than our distributed implementation of the Tarjan-Vishkin shared-memory algorithm
- We are currently exploring optimizations to these approaches.
- **Contact Me:** boglei@rpi.edu

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute under Contract No. DE-AC02-05CH11231 at Rensselaer Polytechnic Institute and Sandia National Laboratories and through the SciDAC ProSPect project at Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Conclusions and Future Work

- Our parallel ice-sheet propagation algorithm efficiently detects degenerate features of ice-sheet meshes.
- We generalize this algorithm to solve biconnectivity in distributed memory
- This direct generalization is more efficient than our distributed implementation of the Tarjan-Vishkin shared-memory algorithm
- We are currently exploring optimizations to these approaches.
- **Contact Me:** boglei@rpi.edu