A Parallel Graph Algorithm for Detecting Mesh Singularities in Distributed Memory Ice Sheet Simulations

Ian Bogle¹  Karen Devine²  Mauro Perego²  Sivasankaran Rajamanickam²  George Slota¹

¹Rensselaer Polytechnic Institute
Troy, NY
²Sandia National Labs
Albuquerque, NM
Goal: Efficiently Detect Problematic Mesh Features

- Ice sheet simulations fail to converge due to features like hinged peninsulas and icebergs in meshes
- Previous solutions either did not find all classes of features or were not able to be run at each step of the simulation
- New algorithm that detects all problematic features
- Distributed memory implementation provides good strong scaling and weak scaling up to 4096 nodes
- Detection takes at most 0.4% of a simulation step’s runtime
- 46,000x faster than previously used preprocessing on highest resolution meshes
Ice Sheet Simulations are Used to Predict Sea Level Rise

• Sea Level Rise (SLR) is something everyone is worried about

• The largest contributor to SLR is the melting of the ice sheets in Antarctica and Greenland

• To accurately predict SLR we need to accurately model ice sheets
MALI is an Unstructured Mesh Finite Element Land Ice Simulator

- We use the 2D mesh that MALI extrudes for simulations
- Our work aims to increase convergence of the simulations
- Interested in the velocity solving component of MALI

 Courtesy of Matt Hoffman (LANL)
Mesh Singularities are Parts of the Mesh That Can Rotate or Translate

Icebergs

Floating Peninsulas
(Floating Hinges)

Blue ice is floating
Brown ice is on the ground

Any part of the mesh that can freely rotate makes the velocity solution not unique
Mesh Singularities Have at Most One Connection to the Ground

These two vertices allow for two unique connections to ground.

Only one unique connection to ground exists, it is through this vertex.

There are no Mesh Singularities

These are Mesh Singularities

- Blue vertex: that is floating in the water
- Red vertex: that is touching the ground
Mesh Singularities Cause Headaches for Scientists Running Simulations

- Mesh Singularities cause convergence problems for solvers
- Mesh Singularities are difficult to detect
- Mesh Singularities can form during simulation
Detecting/Removing Mesh Singularities is the Most Efficient Approach

- Tuminaro et. al. 2016 shows this
  - Proposes a “quick fix” to find both icebergs and floating peninsulas
  - Implemented in a serial matlab code
- Zou et. al. detect icebergs only, and on a structured mesh
- Harrison et. al. 2015 detects icebergs in 3D meshes
- A well-studied graph problem, Biconnectivity, relates to this problem
  - Work-optimal serial algorithm by Hopcroft and Tarjan 1973
  - Shared-memory parallel algorithm by Tarjan and Vishkin 1985
  - Two shared-memory algorithms by Slota et. al. 2014
  - A more recent shared-memory algorithm by Chaitanya et. al. 2016
Converting the Mesh into a Graph Loses Useful Information

Meshes have vertices, edges, and elements

Graphs have only vertices and edges

Easy to spot holes in the ice

Difficult to tell where ice is
Detecting Mesh Singularities is Related to Biconnectivity in Graphs

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

This would not solve our problem completely

Articulation Points are vertices that disconnect the graph if removed.
Application Provides a Mesh and Grounding Information

Legend

- Floating
- Touching Ground
We Identify Parts of the Mesh with No Mesh Singularities
Our Approach Propagates Grounding Information Through the Mesh

• Propagating the initial grounding carefully should reveal Mesh Singularities

• We have access to useful information from the application apart from the graph representation of the Ice Sheet.

• Our approach has two steps:
  • Find Potential Articulation Points
  • Propagate Grounding Information

• **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

This is the only actual Articulation Point

Vertices with more than two incident red edges are Potential Articulation Points
Step 2: Propagate Grounding Information

Legend

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

Start with grounding given from the application...
Step 2: Propagate Grounding Information

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
- **Blue**: Floating
- **Brown**: Initially Grounded
- **Yellow**: 1 Path to Ground
- **Green**: 2 Paths to Ground
- **P**: Potential Articulation Point

Stop the propagation at the Potential Articulation Points

![Diagram showing propagation of grounding information](image-url)
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 Our Algorithm

Legend

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌈</td>
<td>Floating</td>
</tr>
<tr>
<td>🍃</td>
<td>Initially Grounded</td>
</tr>
<tr>
<td>🟢</td>
<td>1 Path to Ground</td>
</tr>
<tr>
<td>🟢🟢</td>
<td>2 Paths to Ground</td>
</tr>
<tr>
<td>🟢🟢♡</td>
<td>Potential Articulation Point</td>
</tr>
</tbody>
</table>

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.
Our Algorithm is Provably Correct and Efficient

- We prove that every vertex we label has the correct number of paths
- We prove that every vertex that has a certain number of paths gets labeled correctly
- We show linear expected work complexity $O(|E|)$
  - $|E|$ is the number of edges
- We show linear expected parallel time $O(d)$ on $O(|E|)$ processors
  - $d$ is the graph diameter
- The full proofs are available in the paper
Experimental Setup

• Testing Systems:
  • Amos – 5 rack Blue Gene/Q housed at RPI, with 5k nodes, 80k cores and 80TB RAM
  • Edison – NERSC’s Cray XC30 Supercomputer

• We used a simple block partitioning for our inputs

• We used real quad meshes of the Antarctic Ice Sheet from the ProSPect project, and generated synthetic quad meshes.
Near-Linear Strong Scaling on Real Ice Sheet Mesh

- 1km real mesh
- 13.5 million vertices
- 13.4 million elements
- Ran on up to 4096 nodes
  - One MPI rank per node
Good Weak Scaling on Real Ice Sheet Meshes

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Vertices</th>
<th>Elements</th>
<th>Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>16km (real)</td>
<td>52,465</td>
<td>51,087</td>
<td>1</td>
</tr>
<tr>
<td>8km (real)</td>
<td>210,170</td>
<td>206,436</td>
<td>4</td>
</tr>
<tr>
<td>4km (real)</td>
<td>841,346</td>
<td>831,173</td>
<td>16</td>
</tr>
<tr>
<td>2km (real)</td>
<td>3,368,275</td>
<td>3,341,449</td>
<td>64</td>
</tr>
<tr>
<td>1km (real)</td>
<td>13,479,076</td>
<td>13,413,766</td>
<td>256</td>
</tr>
</tbody>
</table>

![Graph showing solve time in seconds for different number of MPI ranks.](image)
How We Generated Synthetic Ice Sheets

Start with a square Central Ice Mass

Add initial grounding

Add features that have two paths to the Central Ice

Mesh Singularities can be of arbitrary length

Add some Mesh Singularities
Similar Scaling Shows Synthetic Meshes are Realistic

**Synthetic Weak Scaling**

- Time (seconds)
- Number of MPI Ranks
- 3.9K Vtx/Rank

**Synthetic Strong Scaling (16.2 Million Vtx)**

- Time (seconds)
- Number of MPI Ranks
Algorithm Performance is Robust w.r.t. Number of Grounded Vertices

- Vary the percentage of initially grounded vertices
- Real meshes have 89% of vertices initially grounded
Algorithm Performance is Robust w.r.t. Length of Mesh Singularities

• Chain multiple floating peninsulas together

• Real meshes have at most four floating peninsulas chained together
Algorithm Performance is Robust w.r.t. Number of Floating Peninsulas

- Real meshes have between 7 and 912 mesh singularities
- Increasing amount of mesh singularities changes runtime, not scalability
Our Algorithm Scales Well for Realistic Inputs

• Near-linear strong scaling and good weak scaling on real inputs
• Tolerates wide variations in mesh characteristics
Our Algorithm’s Runtime is Negligible w.r.t. Simulation Runtime

- Our algorithm uses at most 0.4% of a simulation step’s runtime
- Low cost relative to application enables dynamic use as mesh evolves

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Elements</th>
<th>Distributed Runtime (MPI Ranks)</th>
<th>Percentage of Simulation Step Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>16km (real)</td>
<td>51,087</td>
<td>0.0176s (6)</td>
<td>0.19%</td>
</tr>
<tr>
<td>8km (real)</td>
<td>206,436</td>
<td>0.0217s (24)</td>
<td>0.158%</td>
</tr>
<tr>
<td>4km (real)</td>
<td>831,173</td>
<td>0.0414s (96)</td>
<td>0.287%</td>
</tr>
<tr>
<td>2km (real)</td>
<td>3,341,449</td>
<td>0.0407s (384)</td>
<td>0.314%</td>
</tr>
<tr>
<td>1km (real)</td>
<td>13,413,766</td>
<td>0.0561s (1536)</td>
<td><strong>0.412%</strong></td>
</tr>
</tbody>
</table>
New Capability Significantly Benefits Ice Sheet Applications

- Avoid expensive pre-processing of mesh
- “Serial Matlab Runtime” is Tuminaro et. al.’s quick fix (2016)
  - Running in serial on a workstation with an Intel Xeon Gold 6146 CPU @ 3.2GHz
- “Distributed” is our approach running on NERSC’s Cray XC30 Supercomputer

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Distributed Runtime (MPI Ranks)</th>
<th>Serial Matlab Runtime</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>16km (real)</td>
<td>0.0176s (6)</td>
<td>1.04s</td>
<td>59x</td>
</tr>
<tr>
<td>8km (real)</td>
<td>0.0217s (24)</td>
<td>5.65s</td>
<td>260x</td>
</tr>
<tr>
<td>4km (real)</td>
<td>0.0414s (96)</td>
<td>34.6s</td>
<td>835x</td>
</tr>
<tr>
<td>2km (real)</td>
<td>0.0407s (384)</td>
<td>245s</td>
<td>6019x</td>
</tr>
<tr>
<td>1km (real)</td>
<td>0.0561s (1536)</td>
<td>2630s</td>
<td>46880x</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

• New algorithm for finding mesh singularities
  • Hinged peninsulas and icebergs in ice sheet meshes
• Enables convergence of iterative solvers in ice sheet simulations
• Demonstrates good scaling on real and synthetic meshes
• Efficient parallel implementation uses at most 0.4% of the runtime of a single simulation step
• Removes need for expensive mesh preprocessing by application
• Investigating generalization to efficient distributed-memory biconnected component detection

• **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

• New algorithm for finding mesh singularities
  • Hinged peninsulas and icebergs in ice sheet meshes
• Enables convergence of iterative solvers in ice sheet simulations
• Demonstrates good scaling on real and synthetic meshes
• Efficient parallel implementation uses at most 0.4% of the runtime of a single simulation step
• Removes need for expensive mesh preprocessing by application
• Investigating generalization to efficient distributed-memory biconnected component detection

• Contact Me: boglei@rpi.edu